Confidence-Based Robot Policy Learning from Demonstration

Sonia Chernova

CMU-CS-09-105

March 5, 2009

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Manuela Veloso, Chair
Christopher Atkeson
Avrim Blum
Cynthia Breazeal (MIT Media Lab)

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

Copyright © 2009 Sonia Chernova

This research was sponsored by DARPA - BBNT Solutions under grant number 9500007812 and 9500008572, Department of the Interior under grant number NBCH1040007, L3 Communication Integrated Systems, L.D., under grant number 4500244745, SRI International under grant number 03-000211, and Lockheed Martin Corporation under grant number 8100001629. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **05 MAR 2009**	2. REPORT TYPE	3. DATES COVERED **00-00-2009 to 00-00-2009**

4. TITLE AND SUBTITLE **Confidence-Based Robot Policy Learning from Demonstration**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University,School of Computer Science,Pittsburgh,PA,15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR’S ACRONYM(S)
	11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**143**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Keywords: learning from demonstration, multi-robot learning, human-robot interaction, robotics

Abstract

The problem of learning a policy, a task representation mapping from world states to actions, lies at the heart of many robotic applications. One approach to acquiring a task policy is learning from demonstration, an interactive technique in which a robot learns a policy based on example state to action mappings provided by a human teacher.

This thesis introduces Confidence-Based Autonomy, a mixed-initiative single robot demonstration learning algorithm that enables the robot and teacher to jointly control the learning process and selection of demonstration training data. The robot to identifies the need for and requests demonstrations for specific parts of the state space based on confidence thresholds characterizing the uncertainty of the learned policy. The robot's demonstration requests are complemented by the teacher's ability to provide supplementary corrective demonstrations in error cases. An additional algorithmic component enables choices between multiple equally applicable actions to be represented explicitly within the robot's policy through the creation of option classes.

Based on the single-robot Confidence-Based Autonomy algorithm, this thesis introduces a task and platform independent multi-robot demonstration learning framework for teaching multiple robots. Building upon this framework, we formalize three approaches to teaching emergent collaborative behavior based on different information sharing strategies. We provide detailed evaluations of all algorithms in multiple simulated and robotic domains, and present a case study analysis of the scalability of the presented techniques using up to seven robots.

Acknowledgements

It is a pleasure to thank the many people who helped me along the way to completing this thesis.

I would like to thank my advisor, Manuela Veloso, who, starting from the time I was an undergraduate, has provided much help, encouragement and guidance. Without her support and the many opportunities she provided, I would not be where I am today. I am also thankful to my entire thesis committee, Chris Atkeson, Avrim Blum and Cynthia Breazeal, for all of their valuable advice and feedback.

I would like to thank Bernardine Dias, who as my Big Sister in Women@SCS many many years ago originally helped me get started in robotics. And a huge thanks to Matt Mason for introducing me to robotics research.

Thank you to Sony for providing the QRIO robots that were used in my thesis research. Special thanks to all the members of the Sony Intelligence Dynamics Lab, and to Ronald Arkin, for making my stay in Japan exciting, fun and very rewarding.

Many thanks to all of my labmates and friends at Carnegie Mellon, especially Colin McMillen, Doug Vail, Liz Crawford, Scott Lenser, Jim Bruce, Brenna Argall, Kristen Stubbs, Katrina Ligett and Dan Licata, for their ideas, advice, time spent listening to practice talks, and many fun dinners. An enormous thanks to Alice Brumley for always being there when I needed any kind of help. Many things would not have been possible without you!

Last but not least, I would like to thank my family. Thank you to my parents for their unwavering support and encouragement, and for being great role models. And innumerable thanks to my husband James, and our son Alex, who both changed my life in wonderful ways.

Table of Contents

1 Introduction **1**

1.1 Approach . . . 2

1.1.1 Single-Robot Learning . . . 3

1.1.2 Multi-Robot Learning . . . 5

1.1.3 Evaluation . . . 6

1.2 Contributions . . . 7

1.3 Document Outline . . . 8

2 Confidence-Based Autonomy **11**

2.1 Definitions . . . 11

2.2 Confidence-Based Autonomy . . . 12

2.3 Human-Robot Interaction . . . 14

2.4 Overview of Teaching Process . . . 16

2.5 Chapter Summary . . . 17

3 Confident Execution 19

3.1 Algorithm Description 20

3.2 Distance Threshold 22

3.3 Confidence Threshold 23

3.3.1 Single Fixed Threshold 23

3.3.2 Multiple Adjustable Thresholds 27

3.4 Experimental Evaluation in Driving Domain 30

3.4.1 Domain Description 30

3.4.2 Evaluation Results 32

3.5 Chapter Summary 34

4 Corrective Demonstration 37

4.1 Algorithm Description 37

4.2 Experimental Evaluation in Driving Domain 39

4.3 Chapter Summary 43

5 Learning Equivalent Action Choices from Demonstration 45

5.1 Problem Definition 47

5.2 Option Classes 48

5.3 Experimental Evaluation 51

5.3.1 Obstacle Avoidance Domain 51

5.3.2 Evaluation Results 52

5.4 Chapter Summary 54

6 Multi-Robot Learning and Collaboration 55

6.1 Teaching Multi-Robot Collaboration 56

6.1.1 Implicit Coordination without Communication 57

6.1.2 Coordination through Active Communication 58

6.1.3 Coordination through Shared State 58

6.1.4 Discussion 59

6.2 Ball Sorting Domain 59

6.3 Ball Sorting Example 61

6.4 Evaluation Results 63

6.4.1 Implicit Coordination without Communication 63

6.4.2 Coordination through Active Communication 64

6.4.3 Coordination through Shared State 66

6.4.4 Combined Approach and Comparison 68

6.5 Heterogeneous Robot Teams 72

6.5.1 Playground Domain 73

6.6 Chapter Summary 74

7 Scalability Analysis 77

7.1 Beacon Homing Domain 78
7.2 Evaluation 80
7.2.1 Synchronous Learning Start Times 80
7.2.2 Offset Learning Start Times 86
7.2.3 Common Policy Learning 87
7.3 Chapter Summary 89

8 Related Work **91**

8.1 Characterization of Demonstration Learning Approaches 91
8.1.1 Data Gathering 92
8.1.2 Policy Learning 95
8.2 Multi-Robot Learning 98
8.3 Multi-Robot Control 99

9 Conclusion and Future Work **101**

9.1 Contributions 101
9.2 Future Directions 103

A Confidence-Based Autonomy Software System **107**

A.1 Graphical User Interface 109
A.2 LCI Execution Modes 110

B RoboCup Standard Platform League Robot Soccer 113

List of Figures

1.1 The Sony AIBO ERS-7 and QRIO robots. 6

1.2 Diagram of thesis organization. 10

2.1 Confidence-Based Autonomy: overview of the Confident Execution and Corrective Demonstration learning components. 13

2.2 Screenshot of the LCI graphical user interface. 15

2.3 Level of robot autonomy over the course of training, averaged over multiple trials. 16

3.1 Examples of state regions targeted by demonstration requests. 20

3.2 The corridor navigation domain. 24

3.3 Corridor navigation domain training data. 25

3.4 Examples of fixed threshold failure cases 26

3.5 Autonomy threshold calculation 28

3.6 Multiple adjustable thresholds applied to the failure cases shown in Figure 3.4. 28

3.7 Classification of dataset into high and low confidence regions using different classification methods. Table below the figures presents the accuracy and threshold values for each learning method. 29

3.8 Highway driving domain. 31

3.9 Evaluation and comparison of policy learning using Teacher Guided and Confident Execution learning methods. 33

4.1 Evaluation and comparison of policy learning using Corrective Demonstration and Confidence-Based Autonomy learning methods. 40

4.2 Confidence-Based Autonomy demonstrations over time. 41

5.1 Examples of complete and incomplete policies. 46

5.2 Option class policy obtained from the inconsistent demonstration distribution in Figure 5.1(b). 49

5.3 Obstacle avoidance domain and data library. 52

5.4 Three example option class policies for the obstacle avoidance domain. . . . 53

6.1 Ball sorting domain. 60

6.2 Collaborative ball sorting example. 62

6.3 Example Task 2 learning sequence using coordination through active communication. 65

6.4 Example Task 2 learning sequence using coordination through shared state. . 67

6.5 Playground domain. 72

7.1 Beacon homing domain 79

7.2 Average level of autonomy of a single robot over the course of training. . . . 81

7.3 Average number of demonstrations performed by the teacher for each robot and in each experiment. 81

7.4 Total training time with respect to number of robots. 82

7.5 Attention demand on the teacher. 83

7.6 Distribution of the number of simultaneous requests for each experiment. . . 84

7.7 Average demonstration delay. 85

7.8 Comparison of the synchronous and offset learning start time approaches. . . 87

7.9 Comparison of the single common policy and multiple individual policy *flexMLfD* learning approaches. 88

A.1 Confidence-Based Autonomy demonstration learning software system. 108

A.2 Screenshot of the LCI graphical user interface. 109

A.3 Interaction between the LCI and system components in each execution mode. 111

B.1 Standard Platform League robots. 114

B.2 The field for the 2007 AIBO standard platform league competition. 115

B.3 Example AIBO vision data. 115

B.4 Images of robot soccer. 116

List of Tables

5.1 Comparison of CBA learning performance with and without option classes. . 53

6.1 Representation of Task 3 using active communication, shared state and a combined approach. 70

6.2 Summary of Task 3 evaluation of active communication, shared state and a combined approach. 71

Chapter 1

Introduction

The problem of learning a policy, a task representation mapping from world states to actions, lies at the heart of many robotic applications. One approach to acquiring a task policy is *learning from demonstration*, an interactive learning approach based on human-robot interaction that provides an intuitive interface for robot programming. In this approach, a teacher performs demonstrations of the desired behavior to the robot. The robot records the demonstrations, typically as state to action mappings, and learns a policy imitating the teacher's behavior by generalizing from this data.

Learning from demonstration is an incremental online learning process in which the robot begins with no knowledge about the task, and acquires training data until a fully autonomous policy representing the complete task is learned. In most real-world domains, the number of possible world states is very large, making demonstration of the correct robot action from every possible state impractical. Instead, demonstrations are used to sample the state space, and a policy is learned by generalizing from this data. This thesis contributes an interactive mixed-initiative approach to demonstration learning that enables the robot and teacher to jointly control the learning process and selection of demonstration training data. This algorithm enables the robot to identify the need for and request demonstrations for specific parts of the state space based on confidence thresholds characterizing the uncertainty of the learned policy. The robot's demonstration requests are complemented by the teacher's ability to provide supplementary corrective demonstrations in error cases. In our evaluation, we show that this mixed-initiative approach significantly reduces the number of demonstrations and improves the performance of the learned policy.

Most demonstration learning algorithms rely on demonstration data to represent a consistent mapping from states to actions. In reality, demonstrations are frequently inconsistent due to many factors, such as human error, noise and the existence of multiple equally applicable actions. This thesis contributes an algorithm that enables inconsistent demonstrations to be modelled explicitly within the robot's policy as choices between multiple actions.

Additionally, this thesis introduces the problem of *multi-robot learning from demonstration*, which enables a single person to teach multiple robots at the same time. The greatest advantage of multi-robot systems is that by combining their unique abilities, or by simply extending their coverage, multiple robots are able to perform more complex tasks than a single robot alone. In the context of demonstration learning, multi-robot domains present many challenges. Interaction between the robots and the teacher must allow the teacher to perform demonstrations to each individual robot, while also maintaining awareness of the group as a whole. When working with multiple robots, the teacher is not able to pay full attention to all robots at the same time. Each robot must therefore be tolerant of periods of neglect from the teacher. Finally, most collaborative tasks require robots to coordinate their actions through the exchange of information. The demonstration learning algorithm must support inter-robot communication and provide a means of teaching collaborative elements of multi-robot tasks.

This thesis introduces a task and platform independent multi-robot demonstration learning framework that enables a single person to teach multiple robots to perform collaborative tasks. Our approach takes advantage of the adjustable robot autonomy provided by our single-robot algorithm to enable each robot to learn a unique task policy. Building upon this framework, we formalize three approaches to teaching emergent collaborative behavior based on different information sharing strategies. We provide detailed evaluations of all algorithms in multiple simulated and robotic domains, and present a case study analysis of the scalability of the presented techniques using up to seven robots.

1.1 Approach

This thesis seeks to answer the following questions:

- **How can a single robot be taught tasks through human demonstration by enabling both the robot and the teacher to select demonstration examples?**
- **How can the above approach be applied to teaching collaborative multi-robot tasks from demonstration?**

We present our approach to the thesis questions by breaking them further into several subquestions.

1.1.1 Single-Robot Learning

How can the teacher convey task information through demonstration?

An essential part of any demonstration-based learning system is the ability for the teacher to convey task knowledge to the robot. Many different methods for performing demonstrations have been proposed, with most approaches falling into one of two categories:

- *learning from observation* – demonstration examples are obtained by passively observing the actions of the teacher. This approach requires the robot to identify the actions of an external teacher and to map them to its own actions with a corresponding effect.
- *learning from experience* – demonstrations are performed by letting the robot experience the desired actions using its own body, either through manipulation of robot parts or by instructing the robot which of its actions to perform.

Both of the above approaches, which are discussed in detail in Chapter 8, result in a dataset of demonstration examples that enables the robot to learn an action policy. In this thesis, we utilize the learning from experience approach, which has the advantage that the demonstrations are inherently restricted to the robot's physical abilities and a mapping from the teacher's actions to those of the robot is not required. The teacher does not perform demonstrations with her body. Instead, we introduce a *task-independent* and *robot-independent* demonstration interface which allows the teacher to select among the robot's available actions. We assume the robot has a discrete number of of high-level actions, or *action primitives*, which are the basic building blocks required to perform the task. This ensures that the teacher's demonstrations always fall within the robot's abilities.

The robot's perception of the world differs from that of a human. As a result, the teacher's view of the world can differ significantly from that of the robot, either missing some information (communicated data) or perceiving more than the robot can (using information not observable by the robot's sensors). To enable the teacher to perform correct demonstrations, the robot's state information is made available through the control interface display.

When should demonstrations occur?

In addition to selecting a method of demonstration, it is important to determine *when* demonstrations are to be performed. One common assumption held by many demonstration learning approaches is that the teacher, as an expert at the task, always knows when and which demonstrations are required to improve robot performance based on observations of the robot's behavior. However, the robot and teacher represent knowledge in different ways. As a result, the teacher does not always know what additional information the robot requires. In fact, evaluation performed as part of this thesis shows that the teacher is likely to perform redundant demonstrations that provide little information for improving the policy.

In this thesis, we introduce a *mixed-initiative* learning algorithm, Confidence-Based Autonomy (CBA), that enables both the robot and the teacher to select demonstrations. The Confidence-Based Autonomy algorithm consists of two components. The Confident Execution component enables the robot to select demonstrations in real time as it interacts with the environment, using confidence and distance thresholds to target states that are unfamiliar or in which the current policy action is uncertain. To enable the robot to request a demonstration at any time, we assume that the robot can pause between executions of consecutive actions. The Corrective Demonstration component allows the teacher to additionally perform corrective demonstrations when an incorrect action is selected by the robot. The teacher retroactively provides demonstrations for specific error cases instead of attempting to anticipate errors ahead of time. Combined, these techniques provide a fast and intuitive approach for policy learning, incorporating shared decision making between the learner and the teacher.

How can a single robot use the demonstrations to learn a task policy?

Demonstrations provide the robot with a dataset consisting of state-action pairs representing examples of the desired behavior. The robot's goal is to use this information to learn a policy, which enables the robot to select an action based upon its current world state. In selecting a policy learning algorithm we consider a number of requirements:

- Our policy should map from the robot's state, as represented by a set of noisy discrete and continuous sensor values, to a discrete set of action primitives.
- In many domains, it is impossible for the teacher to demonstrate the correct action for every possible state. The policy learning algorithm must therefore generalize over similar but undemonstrated states.
- Due to the interactive nature of learning from demonstration, policy learning must occur in real time.

Based on these requirements, we represent the robot's policy by a classifier mapping from the robot's state to its actions. All algorithms presented in this thesis are classifier-independent, allowing any classifier with a measure of classification confidence to be used.

Additionally, in this thesis we highlight the challenge presented by *equivalent action choices* – situations in which multiple actions are equivalently applicable. Human demonstrators faced with a choice of equivalent actions typically do not perform demonstrations consistently, instead selecting among the applicable actions arbitrarily each time the choice is encountered. As a result, training data obtained by the robot lacks consistency, such that identical, or nearly identical, states are associated with different actions. The resulting inconsistently labeled training data poses a problem for classification-based demonstration learning algorithms by violating the common assumption that for any world state there exists a single best action. An additional contribution of this thesis, is an algorithm that identifies regions of the state space with conflicting demonstrations and enables the choice between multiple actions to be represented explicitly within the robot's policy.

1.1.2 Multi-Robot Learning

How can the teacher perform demonstrations for multiple robots?

One of the problems a teacher faces when working with multiple robots, is the problem of *limited human attention* – the fact that the teacher is not able to pay attention to, and interact with, all robots at the same time. Our approach takes advantage of the adjustable autonomy provided by the single-robot algorithm to enable a single teacher to work with multiple robots at the same time. Controlled by its independent instance of the algorithm, each robot will act autonomously only when highly confident in its actions, and pause to wait for a demonstration in low confidence states. Each robot acquires its own set of demonstrations and learns its individual, possibly unique, task policy.

How can multiple robots be taught to collaborate and work together?

Multiple robots often need to work together to perform complex tasks. In this thesis, we explore techniques for teaching distributed autonomous robots to coordinate their actions, with and without communication. Our approach to teaching collaborative behavior through demonstration relies on *emergent multi-robot coordination* [64], in which the solution to the shared multi-robot task emerges from the complementary actions performed by robots based on their independent policies. We formalize three approaches to teaching collaborative behavior based on different information sharing strategies.

1.1.3 Evaluation

We use several simulated and real-world experimental domains to evaluate the algorithms presented in this thesis. A complete description of each domain is included with each evaluation.

Two robotic platforms are used across the evaluations, the Sony quadruped AIBO and humanoid QRIO robots, shown in Figure 1.1. The AIBO robot was commercially available from Sony for approximately four years, and has been extensively used in robotics research (e.g. [18, 29, 39, 49, 55, 69, 79]) and robotics education (e.g. [23, 72, 88]). The AIBO ERS-7 is a fully autonomous robot, able to perceive the world with its sensors, perform computation using its 576 MHz onboard processor, and to select and execute actions. The robot has a total of 20 degrees of freedom (DOF), three DOF in each leg and neck, and a total of five additional DOF in the tail, ears and mouth. The camera mounted in the head, with a 208x160 pixel resolution, serves as the robot's main sensor. Secondary sensors include IR sensors in the head, and touch sensors in the feet, head and back, as well as wireless communication capabilities. I began my work with the AIBOs in 2002 through the RoboCup robot soccer competitions [50] and my undergraduate senior thesis in 2003. The robot's autonomy and physical dexterity, combined with its expressive physical design, make it highly suitable for demonstration learning applications. In this thesis, during interaction with the human teacher, the AIBO uses LEDs on its head and back to signal for demonstration requests in addition to communicating information wirelessly to the teacher's computer.

The QRIO humanoid robot is a prototype research platform developed by Sony. The QRIO was never commercially available, and was obtained by our research group for several years through an extended equipment loan agreement.[1] The QRIO is a fully autonomous hu-

---

[1] We thank Sony for generously providing the fully programmable QRIO robot and its low-level control software system.

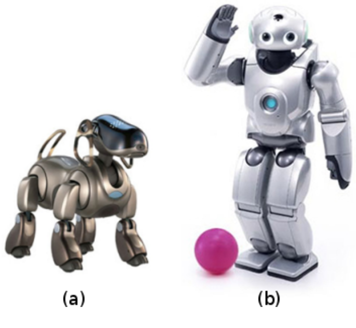


Figure 1.1: The Sony AIBO ERS-7 and QRIO robots.

manoid robot with three 576 MHz onboard processors, wireless communication, 38 degrees of freedom, stereo vision and speech capabilities [44]. The robot's anthropomorphic design and ability to express emotions through human-like motion and speech make it highly suitable for interactive tasks. My work with the QRIO began during an internship at the Sony Intelligence Dynamics Laboratory while working on techniques for providing engaging *long-term* human-robot interaction [17]. We have also used the QRIOs to develop a collaborative commentating behavior for robot soccer games [87]. Within our learning from demonstration work, the QRIO attempts to audibly attract the teacher's attention by speaking the phrase "What should I do now?". Visually, the robot indicates uncertainty by an open arm gesture and the lighting of LEDs on the head.

1.2 Contributions

The contributions of this thesis are as follows:

- Confidence-Based Autonomy, a mixed-initiative single robot demonstration learning algorithm consisting of two components, Confident Execution and Corrective Demonstration. The Confident Execution algorithm enables the robot to learn a behavior policy using a classifier, and to regulate the robot's autonomy based on confidence and distance analysis of the learned classification and the underlying dataset. The Corrective Demonstration algorithm enables the teacher to provide supplementary corrective demonstrations to the robot.
- A method for identifying, representing and enacting equivalent action choices in classifier-based demonstration learning approaches that explicitly models the choice between multiple actions by mapping single label training data to multi-label data classes.
- A task-independent and robot-independent interface for demonstration learning that provides state information to the teacher and enables the selection of demonstration actions.
- Founded on the single-robot demonstration learning approach, a multi-robot learning framework that enables multiple robots to learn independent policies at the same time from a single teacher.
- Three techniques for representing and teaching collaborative multi-robot behavior using demonstration based on different information sharing strategies: implicit coordination, coordination through active communication, and coordination through shared state.

- Evaluation of the scalability of the above multi-robot demonstration learning approach using up to seven quadruped AIBO robots.
- An extensive evaluation of all presented approaches in multiple simulated and real-world domains utilizing multiple robotic platforms.

1.3 Document Outline

The outline below presents a summary of the chapters that follow. Figure 1.2 presents an overview of thesis chapter organization.

Chapter 2 – Confidence-Based Autonomy. We present an overview of our single-robot demonstration learning algorithm, Confidence-Based Autonomy.

Chapter 3 – Confident Execution. We present details of the Confident Execution component of the Confidence-Based Autonomy algorithm, which enables the robot to select demonstrations based on action selection confidence. We evaluate the performance of this algorithm in corridor navigation and highway driving domains.

Chapter 4 – Corrective Demonstration. We present details of the Corrective Demonstration component of the Confidence-Based Autonomy algorithm, which enables the teacher to correct mistakes made by the robot through additional demonstrations. We evaluate the performance of this component, and the complete Confidence-Based Autonomy algorithm, in the highway driving domain.

Chapter 5 – Learning Equivalent Action Choices from Demonstration. We introduce the choice of equivalent actions problem in demonstration learning and present option classes, a method for modeling equivalent action choices explicitly within a discrete classifier-based policy representation. We demonstrate the effectiveness of the option classes approach in an obstacle avoidance domain using a Sony QRIO robot.

Chapter 6 – Multi-Robot Learning and Collaboration. We introduce the problem of multi-robot learning from demonstration and present *FlexMLfD*, a task-independent and robot-independent multi-robot learning framework. Based on this framework, we present three approaches to teaching collaborative multi-robot behavior using different information sharing strategies. We evaluate multi-robot learning and coordination in two domains using Sony AIBO and QRIO robots.

Chapter 7 – Scalability Analysis. We evaluate the scalability of the *FlexMLfD* multi-robot demonstration learning framework in a beacon homing domain using up to seven Sony AIBO robots.

Chapter 8 – Related Work. We discuss the relation of this thesis to previous work in the areas of learning from demonstration and multi-robot learning.

Chapter 9 – Conclusion and Future Work We conclude the dissertation with a summary of its contributions along with a discussion of promising directions for future work based on the presented learning methods.

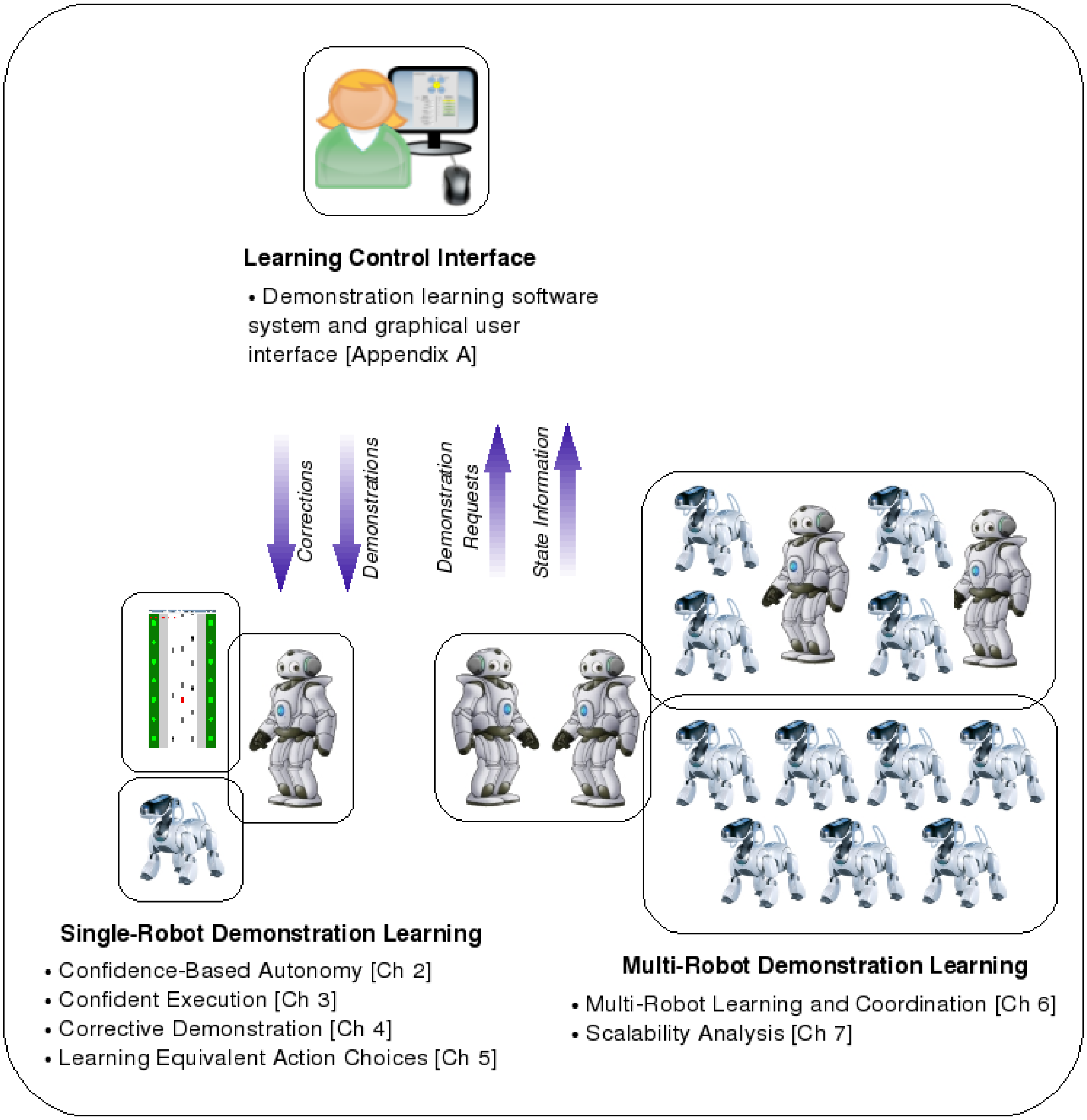

Learning Control Interface
• Demonstration learning software system and graphical user interface [Appendix A]
Corrections
Demonstrations
Demonstration Requests
State Information
Single-Robot Demonstration Learning
• Confidence-Based Autonomy [Ch 2]
• Confident Execution [Ch 3]
• Corrective Demonstration [Ch 4]
• Learning Equivalent Action Choices [Ch 5]
Multi-Robot Demonstration Learning
• Multi-Robot Learning and Coordination [Ch 6]
• Scalability Analysis [Ch 7]

Figure 1.2: Diagram of thesis organization.

Chapter 2

Confidence-Based Autonomy

This chapter presents the Confidence-Based Autonomy algorithm [22] and an overview of the demonstration learning process. We begin with definitions of the notation used throughout this document.

2.1 Definitions

Learning from demonstration approaches rely on training data acquired from demonstrations by a human teacher to learn a task policy imitating the teacher's behavior. Each demonstration is a state-action pair which we assume to represent the correct action to be performed in the given state.

The robot's state, $s = \mathbb{R}^n$, is represented by an n-dimensional feature vector that can be composed of continuous or discrete values. We assume s to be the robot's *observable* state composed of raw or processed sensor data available to the robot; the robot's complete state in the world is usually unknown to it due to sensory limitations.

The robot's actions, a, are bound to a finite set $\mathcal{A}$ of action primitives, which are the basic actions that can be combined to perform the overall task. Given a sequence of demonstrations (s_i, a_i), with state s_i and teacher-selected action $a_i \in \mathcal{A}$, the goal is for the robot to learn to imitate the teacher's behavior by generalizing from the demonstrations and learning a policy mapping from all possible states to actions in $\mathcal{A}$.

We assume the underlying model of the robot's task to be an MDP. The robot's policy is represented and learned using supervised learning based on training data acquired from the demonstrations. Confidence-Based Autonomy can be combined with any supervised learning algorithm that provides a measure of confidence in its classification. The policy is represented by classifier $\mathcal{C} : s \rightarrow (a, c, db)$, trained using state vectors s_i as inputs, and actions a_i as labels. For each classification query, the model returns the highest confidence action $a \in \mathcal{A}$, action selection confidence c, and the decision boundary db with the highest confidence for the query (e.g. Gaussian component for Gaussian Mixture Models (GMMs)). Policy learning for algorithms throughout this thesis is performed using Gaussian Mixture Models or one-against-one multiclass Support Vector Machines (SVM) with a radial basis function kernel [14] and Platt scaling [66], as noted.

2.2 Confidence-Based Autonomy

The Confence-Based Autonomy algorithm enables a human user to train a task policy through demonstration. The algorithm consists of two components:

- *Confident Execution (CE)*: an algorithm that enables the robot to learn a policy based on demonstrations obtained by regulating its autonomy and requesting help from the teacher. Demonstrations are selected based on automatically calculated classification confidence thresholds.
- *Corrective Demonstration (CD)*: an algorithm that enables the teacher to provide supplementary demonstrations and improve the learned policy by correcting mistakes made by the robot during autonomous execution.

Figure 2.1 shows the interaction between the two algorithmic components. Using the Confident Execution algorithm, the robot selects states for demonstration in real time as it interacts with the environment, targeting states that are unfamiliar or in which the current policy action is uncertain. At each timestep, the algorithm evaluates the robot's current state and actively decides between autonomously executing the action selected by its policy and requesting an additional demonstration from the human teacher.

To effectively select demonstrations, the learner must be able to autonomously identify situations in which a demonstration will provide useful information and improve the policy. Confident Execution selects between robot autonomy and a request for demonstration based on the measure of action-selection confidence c returned by the classifier. Given the current state of the learner, the algorithm queries the policy to obtain its confidence in selecting

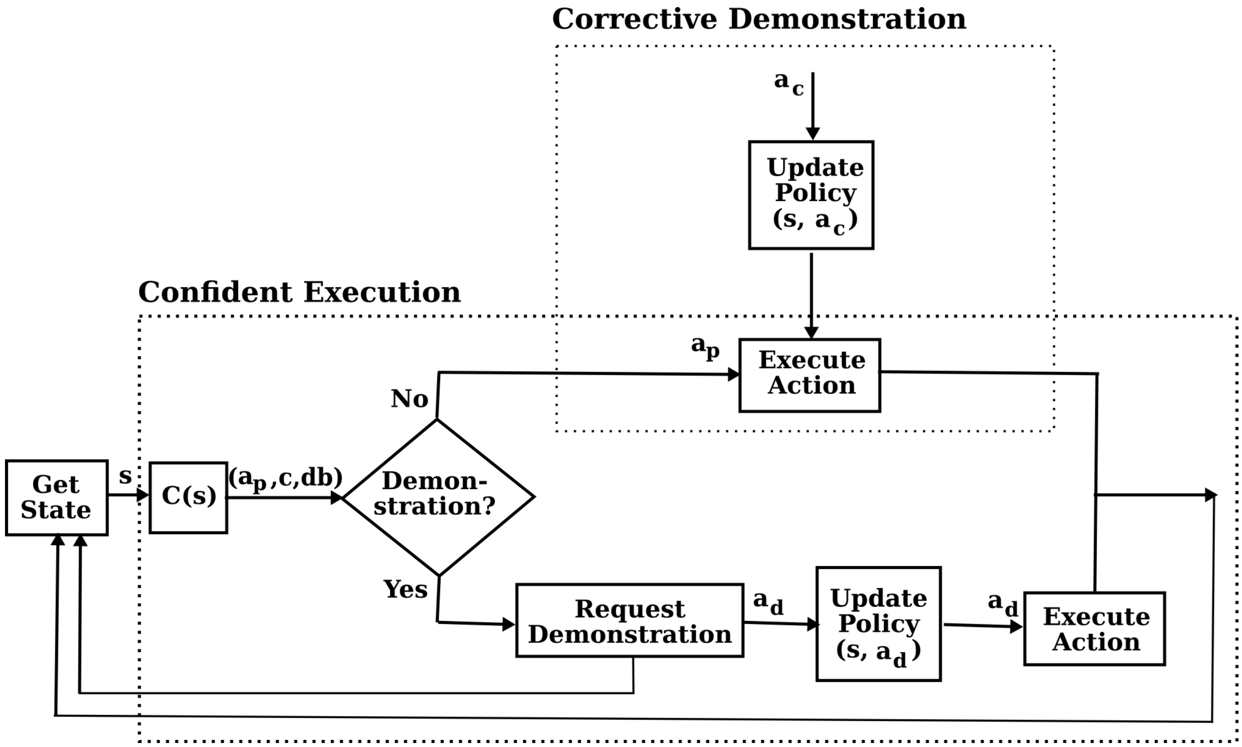

Corrective Demonstration
a_c
Update Policy (s, a_c)
Confident Execution
a_p
Execute Action
No
Get State
s
C(s)
(a_p, c, db)
Demon-stration?
Yes
Request Demonstration
a_d
Update Policy (s, a_d)
a_d
Execute Action

Figure 2.1: Confidence-Based Autonomy: overview of the Confident Execution and Corrective Demonstration learning components.

an action for that state, and regulates its autonomy based on this confidence. The learner executes the returned action a_p if confidence c is above a threshold τ, which is determined by the decision boundary of the classifier, *db*. Confidence below this threshold indicates that the robot is uncertain about which action to take, so it seeks help from the teacher and requests a demonstration. As the robot waits for the teacher's response, it continues to evaluate its current world state and query the policy. This mechanism allows the robot to react to changes in the environment, and possibly regain autonomy if the state changes to one with high action confidence. After receiving demonstration a_d, the algorithm updates the robot's policy using the new training point. As more training data becomes available, the quality of the policy improves and the autonomy of the robot increases until the entire task can be performed without help from the teacher. In Chapter 3 we compare two different methods of using classification confidence to select states for demonstration.

Using the Confident Execution algorithm, the robot incrementally acquires demonstrations as it explores its environment. As it practices its task, the robot uses the policy it learned up to that point to make decisions between demonstration and autonomous execution. However, by relying on the policy before learning is complete, the algorithm is likely to make mistakes due to factors such as overgeneralization of the classifier or incomplete data in some area of the state space. To address this problem we introduce a second algorithmic component, Corrective Demonstration, which allows the teacher to provide corrections for the robot's mistakes. Using this method, when an incorrect action is selected by the algorithm for autonomous execution, the teacher can provide a corrective demonstration indicating which action *should* have been executed in its place. In addition to indicating that the wrong action was selected, this method also provides the algorithm with the correct action to perform in its place, a_c. This type of correction is more informative than negative reinforcement or punishment techniques common in other algorithms, leading the robot to learn quickly from its mistakes. Additionally, the correction can be used to interrupt the execution of the unwanted action in domains for which an interruption will not leave the robot in an unstable state.

Together, Confident Execution and Corrective Demonstration form an interactive learning algorithm in which the robot and human teacher play complementary roles. Using the Confident Execution algorithm, the robot is able to identify states in which demonstration is required; in fact, our results show that the algorithm is able to do this better than the human teacher due to differences in perception and representation abilities. The teacher possesses expert knowledge of the overall task, which is applied to performing demonstrations and correcting execution mistakes using the Corrective Demonstration algorithm. Corrections are associated directly with the state in which the incorrect action decision was made, resulting a fast and effective means of tuning policy performance even for rarely encountered states. Complete details of the Confident Execution and Corrective Demonstration algorithms, and their evaluation, are presented in Chapters 3 and 4, respectively.

2.3 Human-Robot Interaction

To enable interaction between the robot and teacher, a graphical user interface (GUI) is used to display the robot's state information, and to provide the teacher with the choice of actions for demonstration. A screenshot of the GUI is shown in Figure 2.2 with labels highlighting different regions of the display. The interface displays system information, such as the identification number or name of the associated robot, and the robot's current state. State features are organized by their source, separated into information observed locally through the robot's sensors and information received through communication from remote sources, such as teammates or external sensors. Communicated data can be especially difficult for the teacher to monitor without the GUI interface. The robot state display therefore provides vital information necessary to ensure that the teacher's view of the world matches that of

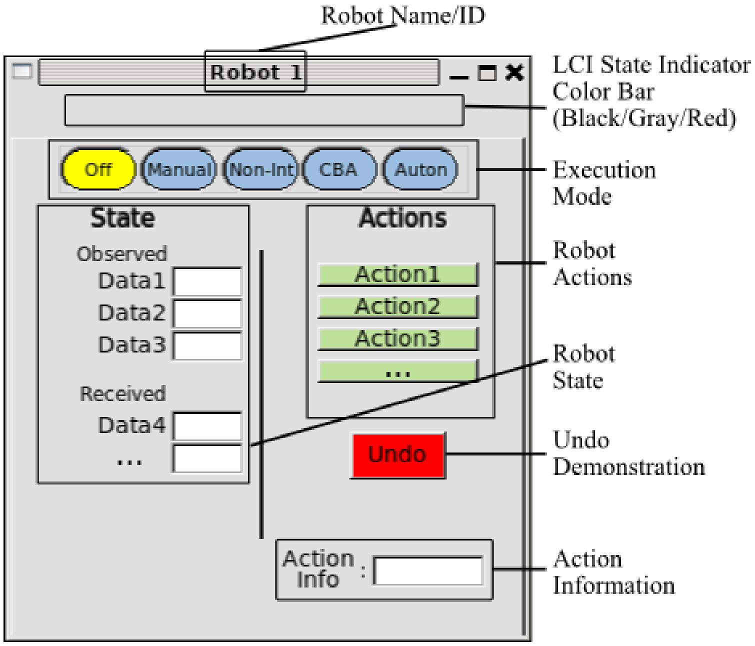

Robot Name/ID
Robot 1
LCI State Indicator Color Bar (Black/Gray/Red)
Off
Manual
Non-Int
CBA
Auton
Execution Mode
State
Observed
Data1
Data2
Data3
Received
Data4
...
Actions
Action1
Action2
Action3
...
Robot Actions
Robot State
Undo
Undo Demonstration
Action Info :
Action Information

Figure 2.2: Screenshot of the LCI graphical user interface.

the robot.

Using the GUI, the teacher is able to perform demonstrations by selecting among a set of possible actions for the robot to execute. The GUI also allows the teacher to undo incorrect demonstrations in the case that the wrong action was accidentally selected. The undo operation erases the last demonstration performed from the policy database. However, it does not undo the effects of the incorrect action that was executed by the robot as a result of the mistaken demonstration. An action information display, located in the lower right corner of the GUI, shows the current action being performed by the robot. When the robot is idle and a demonstration request is pending, the display shows the highest confidence policy action as a recommendation to the teacher.

A color bar at the top of the GUI screen changes color depending upon the current status of the robot: not connected (black), connected and executing an action (gray), or waiting for a demonstration (red). This enables the teacher to identify at a glance when the robot requests a demonstration, or if connectivity to the robot is broken. Finally, a set of buttons at the top of the display control the execution mode of the robot. Complete details of the demonstration learning software and interface are presented in Appendix A.

2.4 Overview of Teaching Process

The following set of steps outline the typical demonstration learning process:

1. Configuration – The teacher configures the CBA learning system by creating an XML configuration file specifying robot identification and connectivity information, as well as selecting the set of state features and actions relevant for the task at hand. Once a connection to the robot is established, task and robot information is displayed in the LCI GUI. Learning is initiated by activating the "CBA" learning mode.

2. Demonstration Learning – The robot begins with no prior knowledge about the task and must request many demonstrations early in the training process. As training data is gathered over time, the quality of the robot's policy improves and its autonomy increases. By practicing the task, the robot refines its policy upon encountering rare or complex states. Learning is complete once the correct action is selected for all states with high confidence and the entire task is performed correctly without help from the teacher.

3. Autonomous Task Execution – Once learning is complete, the robot can use the learned policy to execute the task autonomously.

Figure 2.3 shows how the level of robot autonomy, measured as the percentage of autonomous actions versus demonstrations, changes for over the course of training[1]. The data curve shows

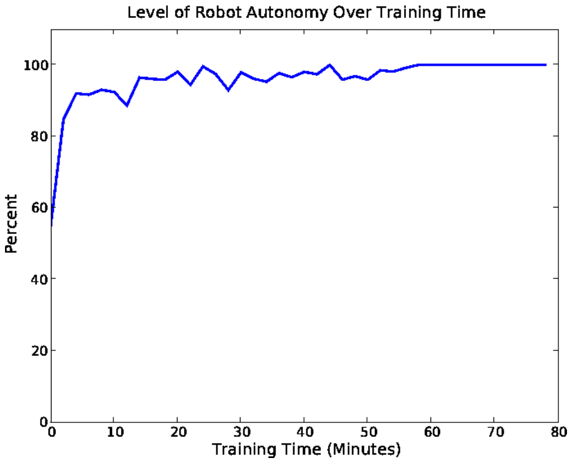


Figure 2.3: Level of robot autonomy over the course of training, averaged over multiple trials.

---

[1] Data in this graph was recorded in the beacon homing domain, see Section 7.1.

that following an initial burst of demonstrations, the robot quickly achieves an average autonomy rate of 80–90%. The robot then continues to refine the policy until full autonomy is reached. The shape of the curve in Figure 2.3 is typical of most CBA learning domains in which initial demonstrations provide the robot with the experience for handling most commonly encountered domain states.

2.5 Chapter Summary

This chapter presented an overview of the Confidence-Based Autonomy algorithm and demonstration learning process. In the next chapter, we present a detailed description of the Confident Execution component and compare its performance to manual demonstration selection by a human teacher. Chapter 4 presents details and evaluation of Corrective Demonstration and the complete Confidence-Based Autonomy algorithm.

Chapter 3

Confident Execution

The Confident Execution algorithm enables a robot to select demonstration states, regulate its autonomy, and learn an action policy based on demonstrations acquired from a human teacher [20, 21]. At each timestep, the algorithm uses thresholds to determine whether a demonstration of the correct action in the robot's current state will provide useful information and improve the robot's policy. If demonstration is required, the robot requests help from the teacher, and updates its policy based on the resulting action label. Otherwise the robot continues to perform its task autonomously based on its policy.

Learning from demonstration is an incremental process in which the robot interleaves learning and execution. In regulating the robot's autonomy, our goal is to prevent autonomous action execution in regions of uncertainty. Specifically, there are two distinct situations in which the robot requires help from the teacher: unfamiliar states and ambiguous states. An unfamiliar state occurs when the robot encounters a situation that is significantly different from any previously demonstrated state, as represented by the outlying points in Figure 3.1. While we do not want to demonstrate every possible state, and therefore need our model to generalize, we would like to prevent over-generalization to truly different states.

Ambiguous states occur when the robot is unable to select between multiple actions with certainty. This situation can result when demonstrations of different actions from similar states, due to sensor noise and other factors, make accurate classification impossible. This is exemplified by the region of overlapping data classes in Figure 3.1. In these cases, we would like the robot to request additional demonstrations to help disambiguate the situation.

The goal of the Confident Execution algorithm is to divide the state space into regions of high confidence (autonomous execution) and low confidence (demonstration) such that unfamiliar

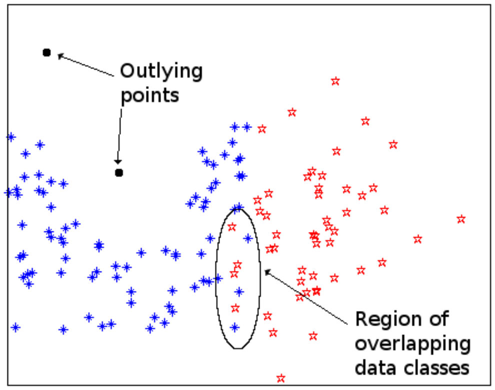


Figure 3.1: Examples of state regions targeted by demonstration requests. Outlying points and regions of overlapping data classes represent unfamiliar and ambiguous state regions, respectively.

and ambiguous regions fall into the low confidence areas. Given a world state, the algorithm uses the following two evaluation criteria to select between demonstration and autonomy:

- *Nearest Neighbor distance*: Given $d = NearestNeighbor(\mathrm{s})$, the distance from the current state to the nearest (most similar) training datapoint, the robot may act autonomously if d is below the distance threshold τ_{dist}.
- *Classification confidence*: Given c, the classification confidence of the current state, the robot may act autonomously if the value of c is above the confidence threshold τ_{conf}.

In the following section, we present the details of the Confident Execution algorithm under the assumption that the values for τ_{dist} and τ_{conf} are known. We then present methods for calculating these thresholds in Sections 3.2 and 3.3.

3.1 Algorithm Description

Algorithm 1 presents a pseudocode description of the Confident Execution algorithm. We assume no preexisting knowledge about the task, and initialize the algorithm with an empty set of training points T. Since a classifier is not initially available, threshold τ_{conf} is initialized

Algorithm 1 Confident Execution Algorithm

```
 1: $T \leftarrow \{\}$
 2: $\tau_{conf} \leftarrow \text{inf}$
 3: $\tau_{dist} \leftarrow 0$
 4: while true do
 5:     if actionComplete then
 6:         $s \leftarrow \text{GetSensorData}()$
 7:         $d = \text{NearestNeighbor}(s)$
 8:         $(a_p, c, db) \leftarrow \mathcal{C}(s)$
 9:         if $c > \tau_{conf}$ and $d < \tau_{dist}$ then
10:             $\text{ExecuteAction}(a_p)$
11:         else
12:             $\text{RequestDemonstration}()$
13:             $a_d \leftarrow \text{GetTeacherAction}()$
14:             if $a_d \neq NULL$ then
15:                 $T \leftarrow T \cup \{(s, a_d)\}$
16:                 $\mathcal{C} \leftarrow \text{UpdateClassifier}(T)$
17:                 $(\tau_{conf}, \tau_{dist}) \leftarrow \text{UpdateThresholds}()$
18:                 $\text{ExecuteAction}(a_d)$
19:     else
20:         //wait for action to complete
```

to infinity to ensure that the robot is controlled through demonstration during the initial learning stage. Distance threshold τ_{dist} is initialized to 0.

The main learning algorithm consists of a loop (lines 4-20), each iteration of which represents a single timestep. The behavior of the algorithm is determined by whether the robot is currently executing an action. If an action is in progress, the algorithm performs no additional computation during this timestep (line 20). Once an action is complete, the algorithm evaluates its state to determine the next action to perform (lines 6-18).

Evaluation begins by obtaining the robot's current state in the environment (line 6). This information is then used to calculate the nearest neighbor distance and to query the learned classifier $\mathcal{C}$ to obtain policy action a_p and confidence c. These values are then compared to the confidence and distance thresholds to decide between demonstration and autonomy (line 9). If similar states have previously been observed, and the learned model is confident in its selection, the algorithm finishes the timestep by initiating the autonomous execution of the policy selected action a_p (line 10). Otherwise it initiates a request for teacher demonstration (lines 12-18).

The robot requests a demonstration by pausing and indicating to the teacher that a demonstration is required. Note that we assume the domain allows the robot to pause execution. Following a demonstration request, the algorithm checks whether a demonstration has been performed (lines 13-14). If the teacher's response is available, a new training datapoint consisting of the current state and the corresponding demonstrated action a_d is added to the training set (line 15). The model classifier is then retrained, and the threshold values updated, before executing the teacher selected action (lines 16-18).

If the teacher's response is not immediately available, the timestep terminates and the whole process is repeated at the next iteration. The robot again senses its state, performs the threshold comparison and checks for a demonstration. This non-blocking mechanism enables the robot to wait for a demonstration from the teacher without losing awareness of its surroundings. In cases where the robot's environment is dynamic, maintaining up to date information is important as the state may change in the time between the initial request and the demonstration. Associating the action label with the robot's most recent state, the one the teacher is most likely responding to, is therefore critical to learning an accurate model. Additionally, changes in the environment can result in the robot attaining a high confidence state without any actions of its own. In these cases, autonomous execution of the task is automatically resumed. In summary, once a demonstration request is made, no further actions are taken by the robot until either a demonstration is received from the teacher, or changes in the environment result in a high confidence state.

Using this approach, Confident Execution enables the robot to incrementally acquire demonstrations representing the desired behavior. As more datapoints are acquired, fewer states distant from the training data are encountered, the performance and classification confidence improve, and the autonomy of the robot increases. Task learning is complete once the robot is able to repeatedly perform the desired behavior without requesting demonstrations. In the following sections we present the methods for calculating the distance and confidence thresholds.

3.2 Distance Threshold

The purpose of the distance threshold is to evaluate the similarity between the robot's current state and previous demonstrations. Our evaluation metric uses the nearest neighbor distance, defined as a the Euclidian distance between a query and the closest point in the dataset. For each robot state query, we obtain its nearest neighbor distance representing the most similar previously demonstrated state. This value is then compared to the distance threshold τ_{dist}.

The value of the distance threshold τ_{dist} is calculated as a function of the average nearest neighbor distance across the dataset of demonstrations. Evaluating the average similarity between states provides the algorithm with a domain-independent method for detecting outliers, points unusually far from previously encountered states. For trials in this article, the value of τ_{dist} was set to three times the average nearest neighbor distance across the dataset.

An alternate method for detecting outliers would be to use classification confidence and request demonstrations in low confidence states. However, situations can arise in which confidence is not directly correlated with state similarity. For example, for many classifiers a set of datapoints encircling an empty region, similar to the shape of a donut, would result in the highest classification confidence being associated with the empty center region far from previous demonstrations. Distance provides a reliable prediction of similarity, even in these cases.

3.3 Confidence Threshold

The purpose of the confidence threshold is to select regions of uncertainty in which points from multiple classes overlap. From the robot's perspective, points in these regions represent demonstrations of two distinct actions from states that appear similar, and are difficult to distinguish based on the sensor data. This problem frequently arises in demonstration learning for a number of reasons, such as the teacher's inability to demonstrate the task consistently, noise in the sensor readings, or an inconsistency between the robot's and teacher's sensing abilities. We would like to set the confidence threshold to a value that prevents either model from classifying the overlapping region with high confidence.

In the following sections, we first present a solution that uses a single fixed confidence threshold value, followed by an algorithm for automatically calculating multiple adjustable thresholds. We then discuss potential drawbacks of the single threshold approach in Section 3.3.1.

3.3.1 Single Fixed Threshold

A single, fixed confidence threshold value provides a direct and simple mechanism to approximate the high confidence regions of the state space. Previous algorithms utilizing classification confidence for behavior arbitration have all used a manually-selected single threshold value, such as 0.5 [52] or 0.25 [35].

In the following section, we present an evaluation of the Confident Execution algorithm utilizing a fixed threshold value in a corridor navigation domain. We compare the learning performance in this task to reinforcement learning. We then discuss drawbacks of the single threshold approach.

Experimental Evaluation and Comparison to Reinforcement Learning

The corridor navigation domain, shown in Figure 3.2, consists of a maze. A Sony AIBO robot must navigate the maze in the circular pattern shown in the figure. The robot observes the environment using the IR sensor built into the head. By turning its head, the robot calculates distance to the nearest obstacle in three directions – left, right and front – resulting in a 3-dimensional continuous feature vector. Due to the noise of the IR sensor, the robot processes all sensor readings when it is stationary and averages the values over 15 consecutive readings.

The robot has four available actions: *forward*, *turn left*, *turn right* and *u-turn*. The *forward* action moves the robot approximately 20 cm in the direction it is facing. *turn left* and *turn right* rotate the robot while slowly advancing it forward, and the *u-turn* action rotates the robot 180°. The teacher demonstrates each action to the robot by making the appropriate selection in the graphical user interface. After completing each action, the robot stops to take the next sensor reading; it requires a minimum of 26 actions to complete one circuit of the domain. The goal of the learning is to classify each observation into one of the four action classes. To perform the task correctly, the robot must learn to distinguish between

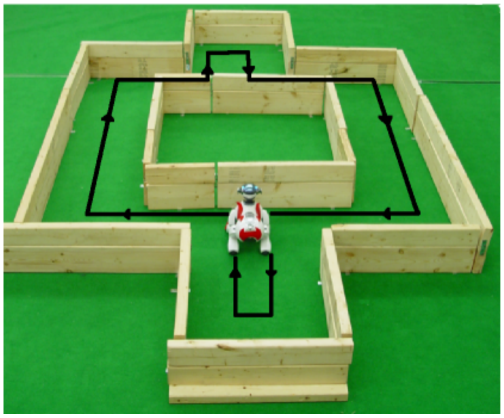
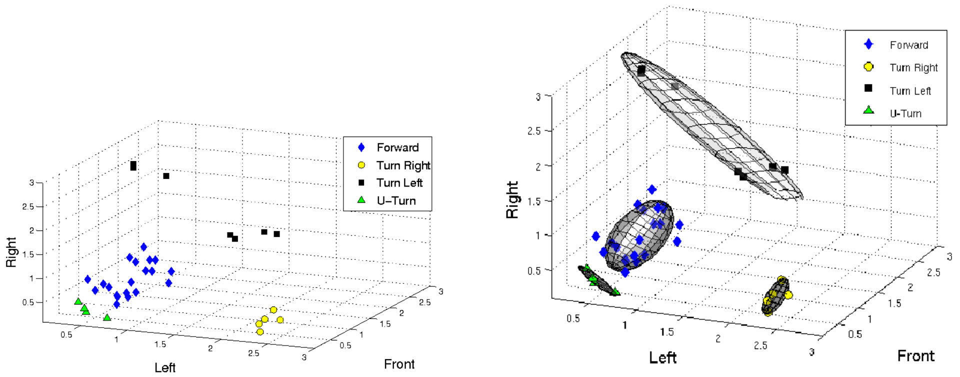


Figure 3.2: The corridor navigation domain. The robot's target path is marked by a black line.

open space, nearby and far away walls.

Using the corridor navigation domain, we compare two policy learning methods, Confident Execution and Prioritized Sweeping [57], a variant of Reinforcement Learning. In the Confident Execution algorithm, the policy was represented by a Gaussian Mixture Model (GMM), and the classification confidence threshold was fixed at two standard deviations for the most likely Gaussian component. Using the Confident Execution algorithm, the robot learned the policy for successfully navigating the corridor domain based on 53 demonstrations performed over the course of five traversals of the domain. After the fifth traversal the robot no longer requested demonstrations, and learning was completed after the robot navigated through the maze autonomously and correctly 10 additional times. Figure 3.3 shows data from the four Gaussian mixture models representing the final learned policy.

Using the Prioritized Sweeping algorithm, the robot learned the task based on reward feedback acquired by exploring the environment. To avoid complex parameters that can affect RL performance in continuous domains, we simplify the domain by discretizing the real-values sensor data into binary wall/no-wall features typical of most grid-world experiments. Our results show that even with this advantageous simplification, RL takes significantly longer to learn the policy than our demonstration-based method while achieving the same performance. Specifically, using a reward function that returns greater reward values for less recently visited states (thereby encouraging circular movement through the domain), the Prioritized Sweeping algorithm took an average of 275 steps, or over 10 complete traversals of the domain, to learn the optimal policy, compared to the 53 demonstrations required by Confident Execution.



(a) (b)

Figure 3.3: (a) Corridor navigation domain training data representing all action classes. (b) Gaussian mixtures fitted to corridor navigation domain training data.

Drawbacks of Using a Single Fixed Threshold

A single fixed threshold value provides a simple control mechanism that is effective for simple domains. However, choosing an appropriate threshold value can be difficult for a constantly changing dataset and model. Figure 3.4 presents examples of three frequently encountered problems.

Figure 3.4(a) presents a case in which two action classes are distinct and fully separable. A model trained on this dataset is able to classify the points with complete accuracy, without misclassifications. However, the current threshold value classifies only 72% of the points with high confidence, marking the remaining 28% of the points as uncertain. In this case, a lower threshold value would be preferred that would allow the model to generalize more freely. The resulting larger high confidence region would reduce the number of redundant demonstrations without increasing the classification error rate of either data class.

Figure 3.4(b) presents an example of the opposite case, in which a stricter threshold value would be preferred. In this example the data classes overlap, resulting in a middle region in which points cannot be classified with high accuracy. A higher threshold value would prevent the classification of points in this region into either data class, initiating instead a request for demonstration that would allow the teacher to disambiguate the situation.

Figure 3.4(c) presents a case in which the datapoints of the two data classes have very different distributions. While the fixed threshold value is appropriate for the left class, 42% of the points in the right class are labeled as low confidence.

Classification of complex multi-class data depends upon multiple decision boundaries. Using

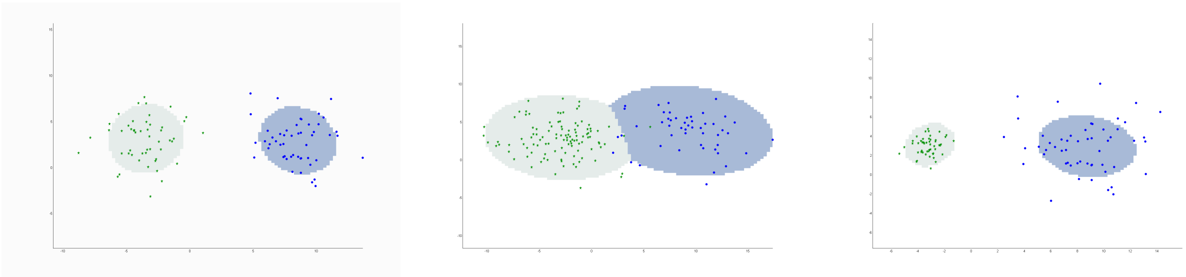

(a) (b) (c)

Figure 3.4: Examples of fixed threshold failure cases: (a) Fully separable data classes with an overly conservative threshold value (b) Overlapping data classes with an overly general threshold value (c) Data classes with different distributions and common threshold value.

the same value for all decision boundaries can exacerbate the problems highlighted above, as a single value often cannot be found that constrains model classification in some areas while allowing generalization in others. The resulting effect is that the robot requests too many demonstrations about things it already knows, and too few demonstrations about unlearned behavior. To address this problem, we present an algorithm for calculating a unique threshold value for each decision boundary.

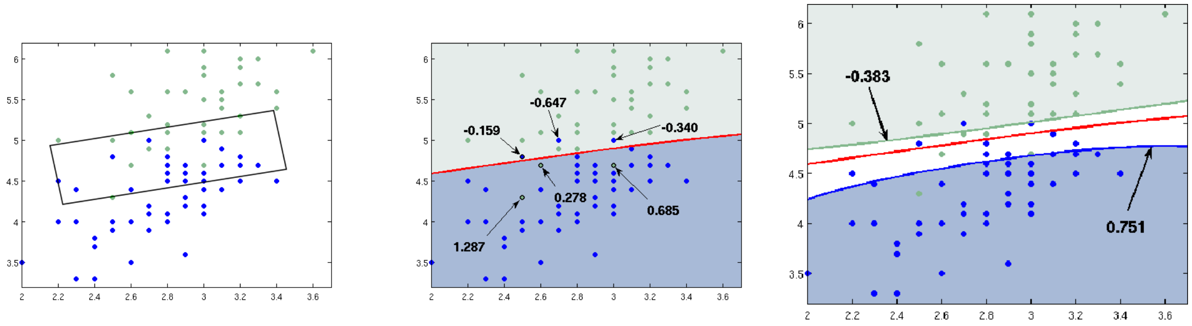
3.3.2 Multiple Adjustable Thresholds

In this section, we contribute an algorithm for calculating a confidence threshold for each decision boundary, customized to its unique distribution of points. In our analysis, we assume that we are able to query the classifier and obtain a confidence score representing the likelihood that a particular input belongs within a specified decision boundary.

The algorithm begins by dividing the dataset into a training and test set and training the classifier $\mathcal{C}$. The resulting learned model is used to classify the withheld test set, for which the correct action labels are known. The algorithm then calculates a unique confidence threshold for each decision boundary based on the confidence scores of misclassified points. Given the confidence scores of a set of points mistakenly classified by a decision boundary, *we assume that future classifications with confidences at or below these values are likely to be misclassifications as well.* The threshold is therefore calculated as a function of these confidence scores.

Specifically, we define a classified point as the tuple (o, a, a_m, c), where o is the original observation, a is the demonstrated action label, a_m is the model-selected action, and c is the model action confidence. Let $M_i = \{(o, a_i, a_m, c) | a_m \neq a_i\}$ be the set of all points mistakenly classified by decision boundary i. The confidence threshold value is set to the average classification confidence of the misclassified points: $\tau_{conf_i} = \frac{\sum^{M_i} c}{|M_i|}$. We take the average to avoid overfitting to noisy data. Other values, based on the maximum or standard deviation, can be used if a more conservative estimate is required. A threshold value of 0 indicates that no misclassifications occurred and the model is able to generalize freely.

Figure 3.5 presents an example of the threshold calculation process. Figure 3.5(a) presents a small sample dataset, the rectangular box in the figure highlights a region of the state space in which points from both classes overlap. Figure 3.5(b) shows the learned decision boundary (in this case a SVM) separating our two data classes. Six misclassified points are marked with the (mis-)classification confidences returned by the model. Misclassified points on each side of the decision boundary will be used to calculate the respective confidence



(a)

(b)

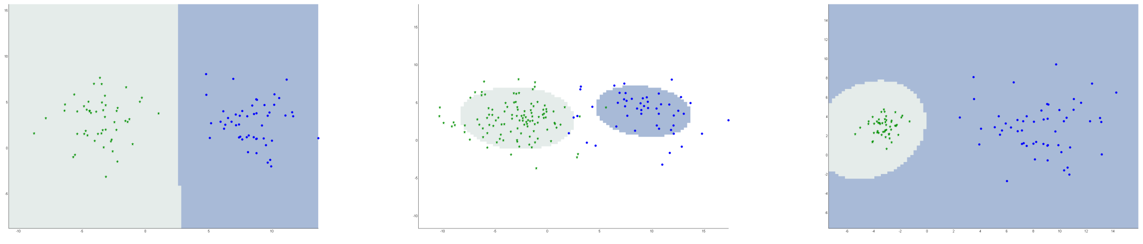
(c)

Figure 3.5: Autonomy threshold calculation: (a) Example dataset, with highlighted overlapping region (b) Learned decision boundary, misclassified points marked with confidence values (c) Learned threshold values for each data class, a low confidence region containing most of the overlapping points remains in the center.

thresholds. Figure 3.5(c) shows the confidence threshold lines and values based on the above calculations. The resulting low confidence region in the middle of the image captures most of the noisy datapoints.

Given this multi-threshold approach, classification of new points is performed by first selecting the action class with the highest confidence for the query. The comparison on line 9 of Algorithm 2 is then performed using the threshold of the decision boundary with the highest confidence for the query. Using this method, the threshold value of the most likely decision boundary to represent the point is used to decide between demonstration and autonomy.

Figure 3.6 shows how the example failure cases discussed in Section 3.3.1 are addressed by the multi-thresholded approach. Customizing the threshold value to each unique data distribution enables the algorithm to correctly classify 100% of the points in Figures 3.6(a)



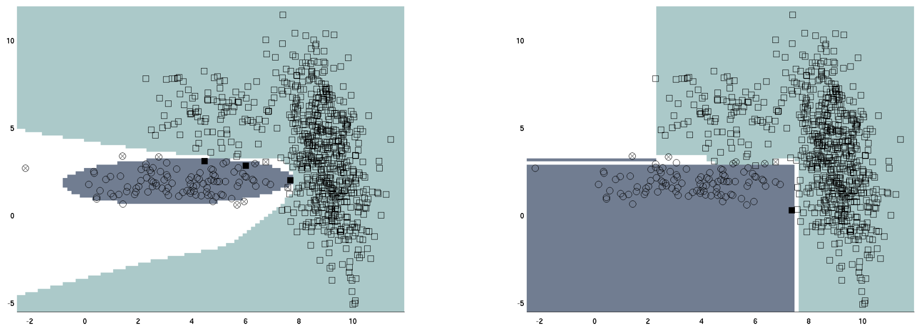
(a)

(b)

(c)

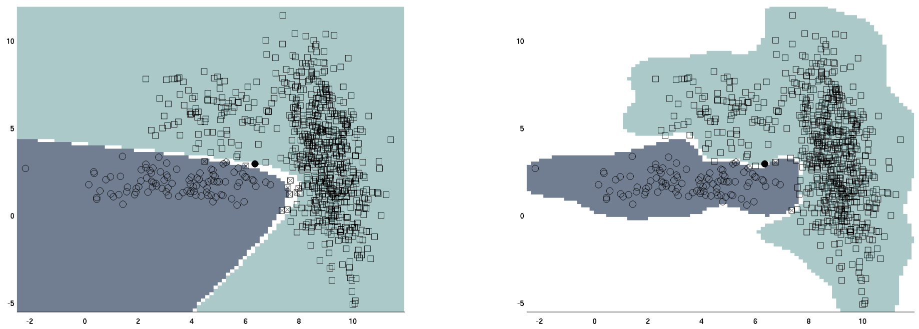
Figure 3.6: Multiple adjustable thresholds applied to the failure cases shown in Figure 3.4.

and (c). Since there are no misclassifications, the model generalizes freely in these examples. For the dataset in Figure 3.6(b), in which perfect classification is not possible, the confidence thresholds are set such that the overlapping region falls into a low confidence area. This example uses a Gaussian mixture model, in which the elliptical confidence gradient around the mean results in a large low confidence area even far from the overlapping region. Other classification methods, such as Support Vector Machines, do not have this drawback.



(a) Gaussian mixture model

(b) Random Forest



(c) SVM (quadratic)

(d) SVM (RBF)

Algorithm	**Correct-Misclas.-Unclass.**	**Thresholds**
GMM	98.6% – 0.4% – 1.0%	(0, 0, 0.012)
RF	99.1% – 0.1% – 0.8%	(0.14, -0.355)
SVM quad.	98.5% – 0.1% – 1.4%	(335.33, -68.77)
SVM RBF	98.9% – 0.1% – 1.0%	(0.825, -0.268)

Figure 3.7: Classification of dataset into high and low confidence regions using different classification methods. Table below the figures presents the accuracy and threshold values for each learning method.

The presented multi-threshold approach is algorithm independent, and Figure 3.7 presents classification results of four different classification methods: Gaussian mixture models, random forests (RF), Support Vector Machine with a quadratic kernel, and SVM with a radial basis function (RBF) kernel. The table below summarizes the classification performance of each algorithm and lists the threshold values for each of the models.

3.4 Experimental Evaluation in Driving Domain

In this section we present an evaluation and comparison of Confident Execution in simulated highway driving domain [1], shown in Figure 3.8.

3.4.1 Domain Description

In the driving domain, the robot represents a car driving on a busy highway. The learner's car travels at a fixed speed of 60 mph, while all other cars move in their lanes at predetermined speeds between 20 and 40 mph. The road has three normal lanes and a shoulder lane on both sides; the robot is allowed to drive on the shoulder to pass other cars, but cannot go further off-road. Since the learner cannot change its speed, it must navigate between other cars and use the shoulder lanes to avoid collision. The robot is limited to three actions: remaining in the current lane, or shifting one lane to the left or right of the current position ($\mathcal{A} = \{\textit{forward,left,right}\}$). The teacher demonstrates the task through a keyboard interface. The simulator has a framerate of 5 fps and is paused during demonstration requests.

The robot's state is represented by: $s = \{l, d_l, d_c, d_r\}$. State feature l is a discrete value symbolizing the robot's current lane number. The remaining three features, denoted by the letter d, represent the distance to the nearest car in each of the three driving lanes (left, center and right). The distance features are continuously valued in the [-25,25] range; note that the nearest car in a lane can be behind the robot. Distance measurements are corrupted by noise to create a more complex testing environment. The robot's policy is relearned each time 10 new demonstrations are acquired.

The driving domain presents a varied and challenging environment; if car distances were to be discretized by rounding to the nearest integer value, the domain would contain over 600,000 possible states. Due to the complexity of the domain, the robot requires a large number of demonstrations to initialize the classifier, resulting in nearly constant demonstration requests early in the training process. To simplify the task of the teacher, we add a short 300

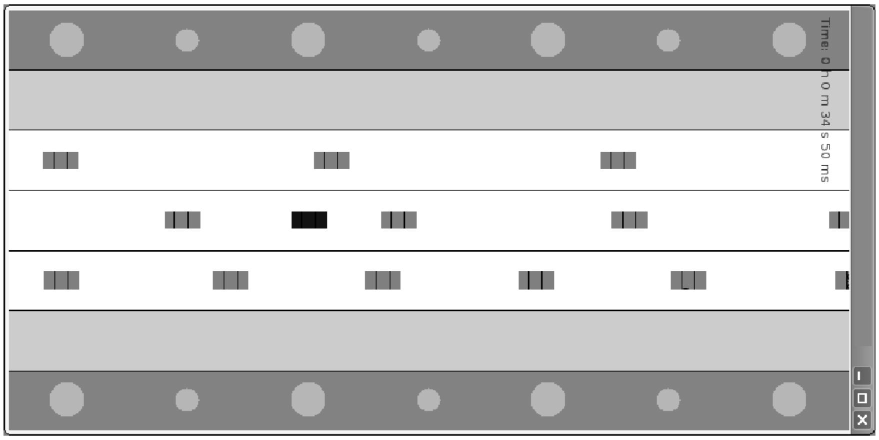


Figure 3.8: Screenshot of the driving simulator. The robot, the black car currently in the center lane, drives at a fixed speed and must navigate around other cars to avoid collisions. The road consists of five lanes: three traffic lanes and two shoulder lanes.

datapoint, or approximately 60 second, non-interactive driving demonstration session to initialize the learning process. While this learning stage is not required, it simplifies the task of the teacher for whom continuous demonstration is preferred over frequent pauses for demonstration requests.

The performance of each learning algorithm was evaluated each time 100 new demonstrations were acquired. For each evaluation, the robot drove for 1000 timesteps over a road segment with a fixed and consistent traffic pattern. This road segment was not used for training, instead each algorithm was trained using a randomly generated car traffic pattern.

Since the algorithm aims to imitate the behavior of the expert, no 'true' reward function exists to evaluate the performance of a given policy. We present two domain-specific evaluation metrics that capture the key characteristics of the driving task. Our first evaluation metric is the robot's *lane preference*, or the proportion of the time the robot spends in each lane over the course of a trial. This metric provides an estimate of the similarity in driving styles. Since the demonstrated behavior attempts to navigate the domain without collisions, our second evaluation metric is the *number of collisions* caused by the robot. Collisions are measured as the percentage of the total timesteps that the robot spends in contact with another car. Always driving straight and colliding with every car in the middle lane results in a 30% collision rate.

3.4.2 Evaluation Results

Using the driving domain, we present the performance evaluation and comparison of the following demonstration selection techniques:

- *TG* – Teacher-guided, all demonstrations selected by the teacher without any confidence feedback from the algorithm
- CE_S – Confident Execution, all demonstrations selected by the robot using a single fixed confidence threshold
- CE_M – Confident Execution, all demonstrations selected by the robot using multiple adjustable confidence thresholds

For each demonstration selection method, the underlying policy of the robot was learned using multiple Gaussian mixture models, one for each action class.

Figure 3.9 presents performance results of the five algorithms with respect to the above defined lane preference and collision metrics. For each evaluation, the figure presents a bar representing a composite graph showing the percentage of time spent by the robot in each lane. The value above the bar indicates the number of demonstrations upon which the evaluated policy is based. The value below the bar indicates the percentage of incurred collisions during the evaluation.

The bar on the right of the figure shows the performance of the teacher over the evaluation road segment. This evaluation indicates that the teacher prefers to drive in the center and left lanes, followed in preference by the left shoulder, right shoulder and right lane. The teacher also successfully avoids all collisions, resulting in a collision rate of 0%. The goal of the learning algorithm is to achieve a driving lane pattern similar to that of the teacher and also without collisions. Note that, as described in the previous section, policy learning was initialized with the same 300-demonstration dataset for all algorithms. This initialization results in identical performance across all algorithms for this initial learning segment.

The top row in Figure 3.9 summarizes the performance of the teacher-guided demonstration selection approach. In this approach, the teacher performed training by alternating between observing the performance of the robot and selecting demonstrations that, in her opinion, would improve driving performance. The teacher selected all training examples without receiving feedback about action selection confidence, and without the ability to provide corrective demonstrations for incorrect actions that were already executed by the

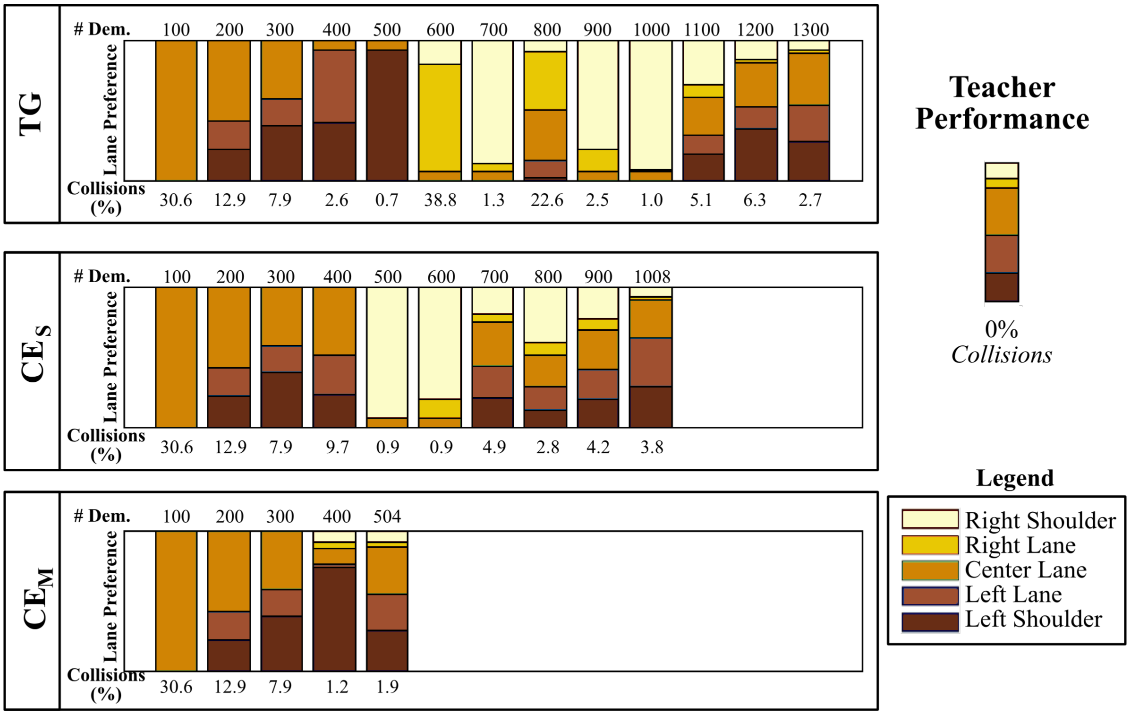


Figure 3.9: Evaluation of the robot's driving performance at 100-demonstration intervals for each of the five demonstration selection methods. The bar graphs indicate the percentage of time the robot spent in each road lane. Values under each bar indicate the percentage of collision timesteps accrued over the evaluation trial. The teacher performance bar on the right of the figure shows the teacher's driving lane preference and collision rate over the evaluation road segment. The goal is for each algorithm to achieve performance similar to that of the teacher.

robot. Instead, the teacher was required to *anticipate* what data would improve the policy. The training process was terminated once the teacher saw no further improvement in robot performance.

Figure 3.9 shows the results of the robot's performance evaluations at 100-demonstration intervals throughout the learning process. The similarity in the driving lane preference of the robot improves slowly over the course of the learning, with significant fluctuations. For example, after 500 demonstrations, the robot's preference is to drive on the empty left shoulder, thereby incurring few collisions. One hundred demonstrations later, the policy has shifted to prefer the center lane. However, the robot has not yet learned to avoid other cars, resulting in a 38.8% collision rate. The policy stabilizes after approximately 1100 demonstrations, representing a driving style similar to that of the teacher, with a small number of collisions. Without confidence feedback from the robot, it is difficult for the teacher to select an exact termination point for the learning. Training continued until, after 1300 demonstrations, the learner's policy showed little improvement. The final policy resulted in a lane preference very similar to that of the expert, but with a 2.7% collision rate.

The second row in Figure 3.9 presents the results of the Confident Execution algorithm with a single autonomy threshold. In this demonstration selection approach, all demonstrations were selected by the robot and learning terminated once the robot stopped requesting demonstrations and performed all actions autonomously. The autonomy threshold value was selected by hand and evaluated in multiple performance trials. Results of the best fixed threshold are presented. Compared to the teacher-guided approach, the policy learned using the CE_S algorithm stabilizes quickly, achieving performance similar to the teacher's after only 700 demonstrations. The number of collisions is again low but persistent, even as the robot gains full confidence and stops requesting demonstrations after 1008 demonstrations. The final lane preference was again similar to that of the expert, with a collision rate of 3.8%.

The third row in Figure 3.9 presents the results of the Confident Execution algorithm with multiple autonomy thresholds, which were calculated using the algorithm presented in Section 3.3.2. Of all the demonstration selection methods, CE_M required the fewest number of demonstrations to learn the task, completing learning after only 504 demonstrations. This result indicates that the use of multiple adjustable thresholds successfully focuses demonstration selection on informative areas of the state space while greatly reducing the number of redundant demonstrations. Throughout the learning process, the number of Gaussian components within the model varied between 9 and 41. This large variation highlights the importance of automating the threshold calculation process, since hand-selecting individual thresholds for each component would be impractical. The lane preference of the final policy was again similar to that of the expert. However, the robot still maintained a small collision rate of 1.9%.

3.5 Chapter Summary

This chapter presented the Confident Execution component of Confidence-Based Autonomy, which enables the robot to select demonstrations in real time as it interacts with the environment using confidence and distance thresholds to target states that are unfamiliar or in which the current policy action is uncertain. Experimentally, we used a complex simulated driving domain to compare three methods of selecting demonstration training data: manual data selection by the teacher, confidence-based selection using a single fixed threshold, and confidence-based selection using multiple automatically calculated thresholds. Based on our evaluation, we conclude that both confidence-based methods were able to select more informative demonstrations than the human teacher. Of the single and multiple threshold approaches, the multiple adjustable threshold technique required significantly fewer demonstrations by focusing onto regions of uncertainty and reducing the number of redundant datapoints.

The evaluation of these three algorithms highlights the difficulty of the driving problem. Each of the approaches was able to select demonstrations that resulted in a policy that mimics the overall driving style of the teacher. However, all of the policies resulted in a small number of collisions, which typically occurred when the robot merged too close to another vehicle and touched its bumper. Such mistakes are difficult to correct using the techniques evaluated so far. Even within the teacher guided demonstration selection method, in which the human teacher has full control of the demonstration training data, by the time the collision has been observed the incorrect decision had already been made by the algorithm. Instead, *retroactive* demonstration is required to correct already made mistakes. In the following chapter, we present the Corrective Demonstration algorithm which enables the teacher to perform such corrections.

Chapter 4

Corrective Demonstration

Many traditional approaches to learning from demonstration rely on an iterative demonstrate-observe strategy in which the teacher alternates between two learning phases, observing robot behavior and providing demonstrations aimed at correcting any observed mistakes [35, 52]. The drawback of these techniques is that not only does the teacher have to guess what demonstrations are required to improve task performance, but she must also recreate the problem case in order to demonstrate the correct behavior. Recreating a rarely encountered state can be challenging in real robotic systems with sensor noise and dynamic environments. This can lead to long training times and many demonstrations, as shown by our evaluation of the teacher-guided demonstration learning method in the previous chapter.

The Confident Execution algorithm, presented in the previous chapter, enables the robot to identify unfamiliar and ambiguous states and prevents autonomous execution in these situations. However, states in which an incorrect action is selected with high confidence for autonomous execution still occur, typically due to over-generalization of the classifier. In this chapter, we introduce the Corrective Demonstration component of the CBA algorithm, which enables the teacher to provide supplementary demonstrations to correct mistakes made by the robot.

4.1 Algorithm Description

The Corrective Demonstration algorithm enables the teacher to observe the autonomous execution of the robot and to correct any incorrect actions *as they occur*, associating the correction immediately with the problem state. Unlike the Confident Execution approach, in which the robot was given the next action to perform, the correction is performed with relation to the past state at which the mistake was made. For example, when observing a driving robot approaching too close behind another car, the teacher is able to indicate that instead of continuing to drive forward, the robot should have been merging into the passing lane. In this way, in addition to indicating that the wrong action was performed, Corrective Demonstration also provides the algorithm with the action that should have been performed in its place. This technique is more effective than negative reinforcement, or punishment, techniques common in other algorithms, leading the robot to learn quickly from its mistakes.

Algorithm 2 combines Corrective Demonstration (lines denoted by $\star$) with Confident Execution and presents the complete Confidence-Based Autonomy algorithm.

Algorithm 2 Confidence-Based Autonomy algorithm, combining Confident Execution and Corrective Demonstration

```
$T \leftarrow \{\}$
$\tau_{conf} \leftarrow \inf$
$\tau_{dist} \leftarrow 0$
while true do
    $s \leftarrow$ GetSensorData()
    if actionComplete then
        $(a_p, c, db) \leftarrow \mathcal{C}(s)$
        $d =$ NearestNeighbor($s$)
        if $c > \tau_{conf}$ and $d < \tau_{dist}$ then
            ExecuteAction($a_p$)
            $s_c \leftarrow s$                                   $\star$
        else
            RequestDemonstration()
            $a_d \leftarrow$ GetTeacherAction()
            if $a_d \neq NULL$ then
                $T \leftarrow T \cup \{(s, a_d)\}$
                $\mathcal{C} \leftarrow$ UpdateClassifier($T$)
                $(\tau_{conf}, \tau_{dist}) \leftarrow$ UpdateThresholds()
                ExecuteAction($a_d$)
    else
        if autonomousAction then                                 $\star$
            $a_c \leftarrow$ GetTeacherAction()                  $\star$
            if $a_c \neq NULL$ then                              $\star$
                $T \leftarrow T \cup \{(s_c, a_c)\}$             $\star$
                $\mathcal{C} \leftarrow$ UpdateClassifier($T$)   $\star$
                $(\tau_{conf}, \tau_{dist}) \leftarrow$ UpdateThresholds()   $\star$
```

The Corrective Demonstration technique comes into play each time the robot executes an autonomous action. As an action is selected for autonomous execution, the algorithm records the robot's state that led to this decision and saves this value within the variable s_c (line 11). During the execution of an autonomously selected action, the algorithm checks for a teacher demonstration at every timestep (lines 22-23). If a corrective demonstration is made, a new training datapoint consisting of the recorded demonstration state s_c and the corrective action a_c is added to the training set (line 24). The classifier and thresholds are then retrained using the new information. Note that once initiated, the incorrect action is allowed to complete without interruption; interrupting an action may cause the robot to enter an unstable or unsafe state.

4.2 Experimental Evaluation in Driving Domain

In this section, we present the performance evaluation and comparison of the following demonstration selection techniques in the driving domain[1]:

- *CD* – Corrective Demonstration, all demonstrations selected by the teacher and performed as corrections in response to mistakes made by the robot
- *CBA* – The complete Confidence-Based Autonomy algorithm combining Confident Execution using multiple adjustable confidence thresholds with Corrective Demonstration

Figure 4.1 presents the results of this evaluation, as well as the results of the Teacher Guided and Confident Execution demonstration selection methods, which were evaluated in the previous chapter, for comparison.

In the fourth row of Figure 4.1 we present our evaluation of demonstration selection using only the Corrective Demonstration algorithm. In this approach, all demonstrations were selected by the teacher as corrections in response to mistakes made by the robot. Behavior corrected by the teacher included collisions, as well as incorrect lane preference (e.g. always driving on the shoulder) and rapid oscillations between lanes. To enable the teacher to accurately perform corrections, the simulation was slowed from 5 to 2 frames per second. Learning was terminated once the robot required no further corrections. As shown in Figure 4.1, the complete training process using Corrective Demonstration took 547 demonstrations, achieving a final policy that correctly imitates the teacher's driving style with a 0% collision rate.

---

[1] See Section 3.4 for a complete description of the domain and experimental evaluation criteria.

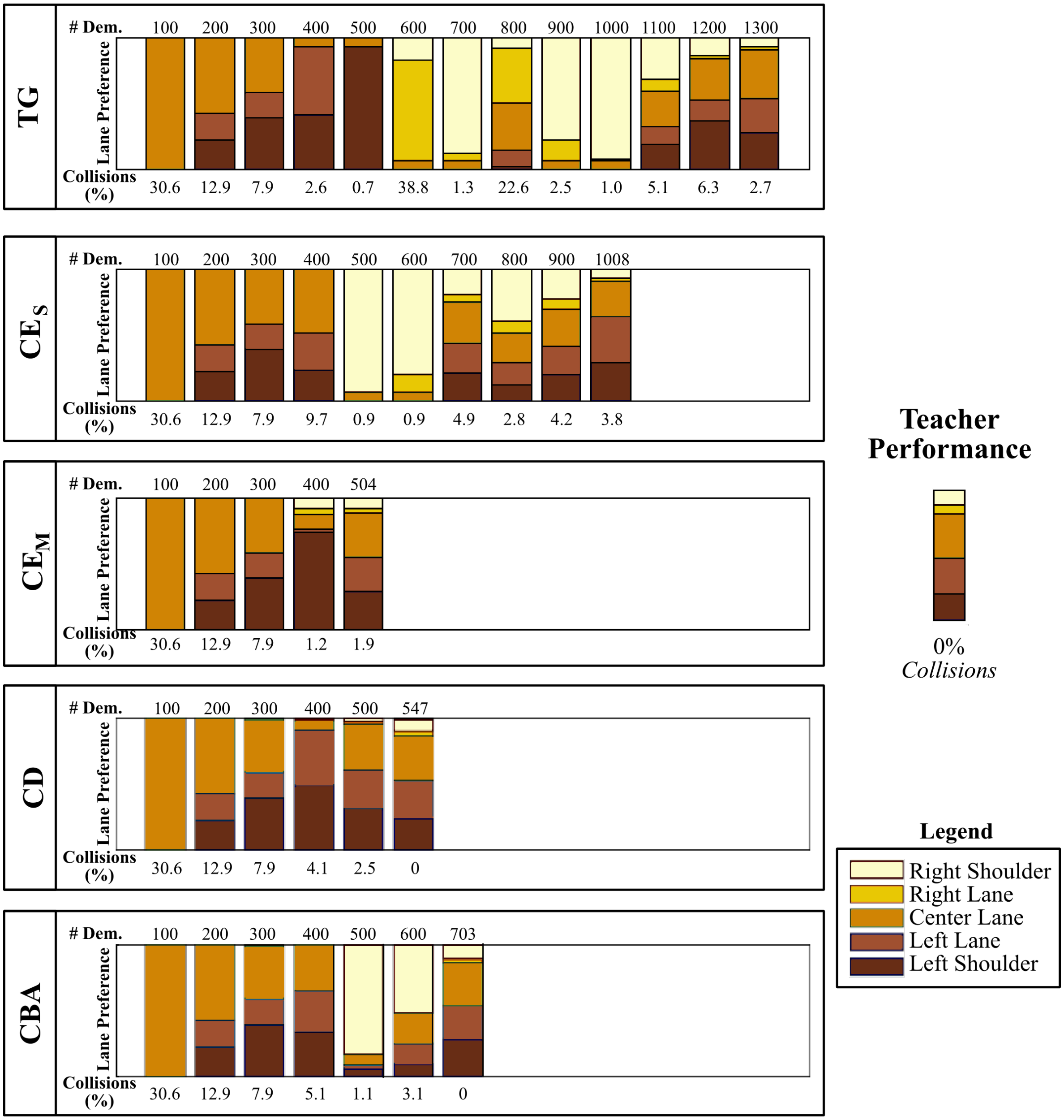
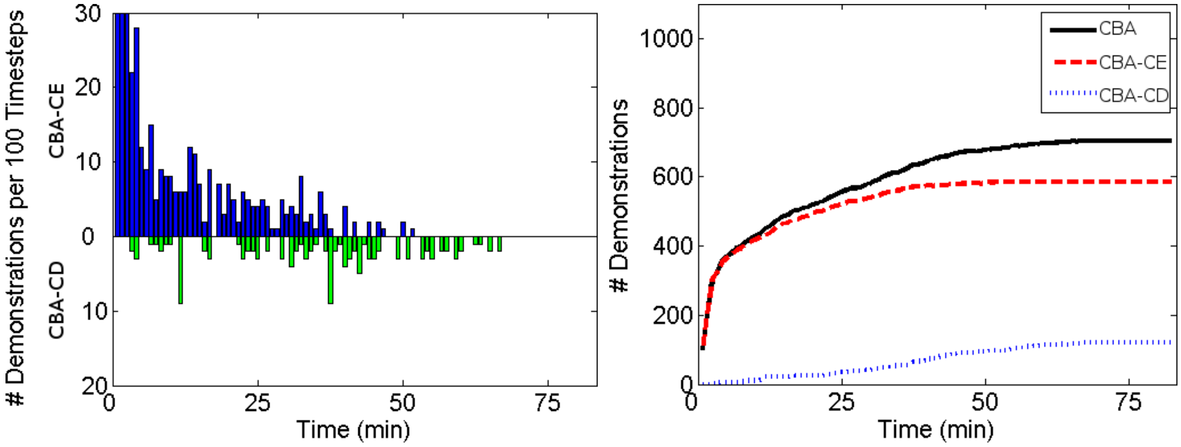


Figure 4.1: Evaluation of the robot's driving performance at 100-demonstration intervals for each of the five demonstration selection methods. The bar graphs indicate the percentage of time the robot spent in each road lane. Values under each bar indicate the percentage of collision timesteps accrued over the evaluation trial. The teacher performance bar on the right of the figure shows the teacher's driving lane preference and collision rate over the evaluation road segment. The goal is for each algorithm to achieve performance similar to that of the teacher.

The final row in Figure 4.1 presents the evaluation of the complete Confidence-Based Autonomy algorithm, which combines CE_M with CD. Using this approach, learning is complete once the robot no longer requests demonstrations *and* is able to perform the driving task without collisions. Using CBA the robot required a total of 703 demonstrations to learn the task, successfully learning to navigate the highway without collisions.

We analyze the impact of the two CBA learning components by comparing the number and distribution of demonstrations acquired by each algorithm during the learning process. In this section we refer to the learning components of CBA as CBA-CE and CBA-CD to differentiate from the algorithm evaluations presented in previous sections. Note that the behavior of the Confident Execution component is dependent upon the method used to set the autonomy thresholds. In this evaluation we use multiple adjustable thresholds calculated as the average value of misclassified points.

In Figure 4.2(a), each datapoint along the x-axis represents the number of demonstrations requested using CBA-CE (top) and initiated by the teacher using CBA-CD (bottom) during a 100-timestep interval, or approximately 40 seconds of simulator runtime (excluding pauses for demonstration requests). Since the first three 100-demonstration timesteps consist entirely of non-interactive demonstration, the values for these timesteps are 100 and, due to scaling, exceed the bounds of the graph. Figure 4.2(b) shows how the cumulative number of demonstrations for each component, and in total, grows with respect to training time. The



(a) (b)

Figure 4.2: (a) Timeline showing how the number of demonstrations initiated by the robot through Confident Execution (top) and initiated by the teacher through Corrective Demonstrations (bottom) changes over the course of the training. (b) The cumulative number of demonstrations acquired by each component, and in total, over time.

complete training process lasts approximately an hour and a half.

Analysis of these graphs shows that most demonstrations occur early in the training process. Importantly, Confident Execution accounts for 83% of the total number of demonstrations, indicating that the robot guides most of the learning. Most of these demonstration requests occur during the first few minutes of training when the robot encounters many novel states and the classification confidence remains low. The robot requires few corrections during this stage because many mistakes are prevented by requesting a demonstration instead of performing a low confidence action. Corrective Demonstration plays its greatest role towards the end of training process, where it accounts for 73% of the final 100 demonstrations. At this stage in the learning the robot's action selection confidence is high enough that it rarely asks for demonstrations. Its policy already closely imitates the teacher's driving style but a small number of collisions remain. Corrective Demonstration enables the teacher to fine-tune the policy and eliminate all collisions. This result highlights the importance of Corrective Demonstration, whether alone or in conjunction with another selection technique, for optimizing policy performance.

While CBA achieves similar final performance compared to the CD algorithm evaluated in the previous section, it requires approximately 150 additional demonstrations to learn this policy. The additional demonstrations can be attributed to Confident Execution demonstration requests that served to increase the classification confidence but did not change the outcome of the robot's action. Viewed another way, these datapoints correspond to states in which the robot would have performed the correct action even if it had not asked for a demonstration. From this result it appears that allowing the robot to make mistakes and correcting them after the fact, as done in the CD evaluation, results in the best demonstration selection approach with respect to the performance metrics defined above and the overall number of demonstrations.

However, taking away the robot's ability to request demonstrations and utilizing only retroactive correction has several drawbacks, namely requiring constant and full attention from the teacher, and, most importantly, requiring the robot to make many mistakes before it learns the correct policy. By comparison, the CBA algorithm enables the robot to request demonstrations in low confidence states, thereby avoiding many incorrect actions. Our original lane preference and collision metrics do not take this difference into account as they focus only on the final policy performance of the robot.

To evaluate the difference between these algorithms, we additionally examine the number of collisions each robot incurs over the course of the learning. This evaluation shows that using the CD algorithm, the robot incurs 48% more collisions (278 vs. 188) during training than by using CBA. As a result, by allowing the robot to request demonstrations in low-confidence states, the CBA algorithm incurs a small increase in the number of demonstrations while greatly reducing the number of incorrect actions performed during learning. The reduction in the number of action errors is significant due to its importance for many learning domains, especially robotic applications in which such errors may pose dangers to the system.

In summary, our evaluation has shown that the ability to retroactively *correct mistakes* is crucial to optimizing the policy and eliminating all collisions. The best performance was achieved by the Corrective Demonstration and Confidence-Based Autonomy methods, with CD requiring fewer demonstrations but incurring a greater number of collisions during training. The choice between CD and CBA can therefore be viewed as a tradeoff between the number of demonstrations and the frequency of undesired actions during training. In fact, CD is a special case of CBA in which the autonomy threshold is set to classify all points with high confidence. Adjusting the selectiveness of the CBA autonomy thresholds could, therefore, provide the user with a sliding control mechanism that effects the robot's tendency to perform autonomous actions versus demonstration requests. Importantly, we note that the overall number of demonstrations required by either approach is less than the teacher-guided method and only a tiny fraction of the overall state space.

4.3 Chapter Summary

This chapter presented the Corrective Demonstration component of the Confidence-Based Autonomy algorithm, which enables the teacher to additionally perform corrective demonstrations when an incorrect action is selected by the robot. Using this approach, the teacher retroactively provides demonstrations for specific error cases instead of attempting to anticipate errors ahead of time. Experimentally, we evaluated the performance of Corrective Demonstration and the complete Confidence-Based Autonomy algorithm in the highway driving domain, and compared this performance to the Teacher Guided and Confident Execution methods presented in the previous chapter. Of the presented approaches, the best policy performance was achieved by the Corrective Demonstration and complete Confidence-Based Autonomy algorithms, both of which achieved a lane preference similar to that of the teacher without any collisions. While Corrective Demonstration required slightly fewer demonstrations to learn the final policy, compared to CBA it resulted in a significant increase in the number of errors made by the robot over the course of the learning process. The CBA algorithm, therefore, provides the best demonstration selection method for domains in which incorrect actions are not desirable during the training process.

Chapter 5

Learning Equivalent Action Choices from Demonstration

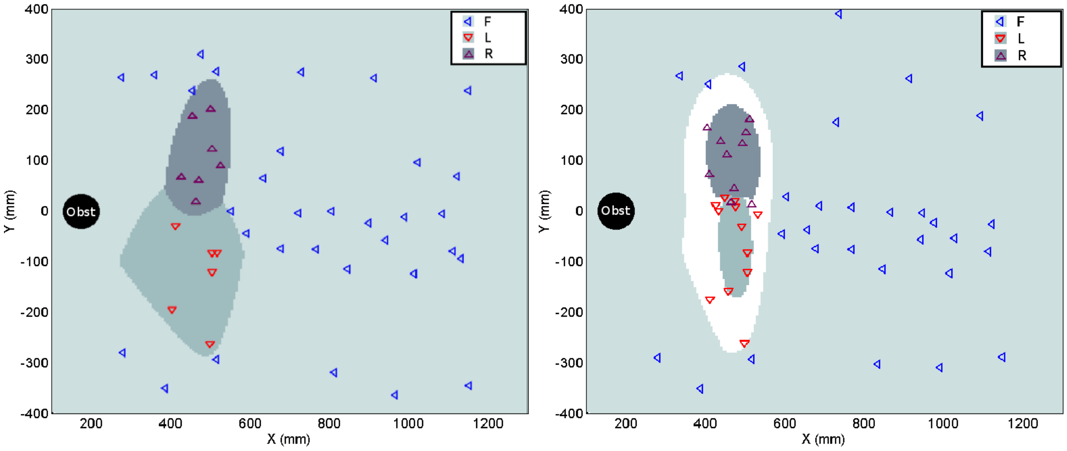
In the previous chapters, we introduced the Confidence-Based Autonomy algorithm and its components, and demonstrated its effectiveness in improving policy performance and reducing the number of demonstration in comparison to manual demonstration selection by a human teacher. The Confidence-Based Autonomy algorithm makes two assumptions about the task policy and demonstration training data: 1) that for each state there exists a single best action, and 2) that the teacher is able to demonstrate this state-action combination consistently. The CBA algorithm relies on these assumptions to learn a *complete* policy given enough demonstration data, where we define a complete policy to be one that classifies the entire state space with high confidence and results in full robot autonomy.

For robots operating in real-world environments, the assumptions of consistent demonstrations and one-to-one state to action mappings can fail, resulting in regions of inconsistent demonstrations in which similar states lead to different actions. One reason for inconsistent demonstrations is that robots often encounter *equivalent action choices*, situations in which multiple actions are equivalently applicable. For example, a moving robot that encounters an obstacle directly in its path has the option of moving left or right to avoid it. If the space is empty, both directions are equally valid for performing the desired task. Furthermore, consider the case in which in addition to deciding the direction of motion, the teacher must also specify the distance at which to begin to avoid the obstacle. Should avoidance actions be initiated at a distance of 1.0 meters, 0.5 meters, or somewhere in between? In reality, a whole range of distances are likely to provide the desired behavior, and over this range multiple actions are equally valid (move forward a little more, move left or move right). Similar choices can arise in many other situations. Human demonstrators faced with a choice of equivalent actions typically do not perform demonstrations consistently, instead selecting among the applicable actions arbitrarily each time the choice is encountered. The result is that the teacher's demonstrations are likely to be inconsistent.

Another common reason for inconsistent demonstration is sensor noise. Consider, for example, the same obstacle domain in which the teacher must decide the direction of motion based on the position of the robot. Even if the teacher selects actions with perfect consistency, these demonstrations may still appear inconsistent to the robot due to noisy readings from the robot's sensors. Finally, human error can be a further source of such data.

Despite the teacher's best intentions, inconsistent demonstrations are very likely to occur. Critically, all inconsistent demonstrations appear the same from the robot's perspective, whether they are the result of a choice of multiple equivalent actions, sensor noise or some other causes. In all cases, these regions are characterized by training data in which identical, or nearly identical, states have different action labels.

Figure 5.1 presents an example of the effect of inconsistent demonstration on Confidence-Based Autonomy policy learning. The images in this figure show two example distributions obtained from demonstration learning in the obstacle avoidance domain (Section 5.3.1). Each point in the figure represents a demonstration performed when the robot was at that (x,y) position. Background shading represents the most likely action class, with white representing



(a) Consistent demonstrations lead to a complete policy.

(b) Inconsistent demonstrations result in non-separable data classes and an incomplete policy with low confidence regions (white).

Figure 5.1: Examples of complete and incomplete policies.

low-confidence (demonstration) regions.

In Figure 5.1(a), consistent demonstrations result in a complete, fully autonomous policy with no low-confidence regions. In Figure 5.1(b), inconsistent demonstrations result in an incomplete policy in the middle region of the graph, where data points from multiple classes overlap. As the robot repeats the task, all states that fall within the low confidence, white, region trigger additional demonstrations. However, unlike the consistent demonstration case, acquiring additional data points will not help the classifier. Instead, an infinite number of overlapping demonstration points would be collected by the algorithm.

Our demonstration learning algorithm aims to learn a policy imitating the behavior of the teacher. If the teacher's demonstrations are consistent, mapping all similar states to a single action, then our goal is for the robot to deterministically execute this action from all such states. If, however, the teacher's demonstrations are inconsistent, leading the robot to observe demonstrations of multiple possible actions from similar states, then our goal is for the robot to select its action *probabilistically* from among the presented examples. To this end, when the robot obtains demonstrations of different actions from similar states, we assume that all these actions are valid and the robot's behavior can be selected at random from among the presented choices. In the following section we formally define this classification problem. We then present an algorithm for identifying regions of the state space in which data from multiple classes overlaps as a result of inconsistent demonstrations, and for modeling the choice between multiple actions explicitly within the robot's action policy as *option classes.* Our automated approach does not require the teacher to predefine or demonstrate special choice actions, extending instead from the person's natural demonstration technique.

5.1 Problem Definition

Traditional classification methods are single-label algorithms that map each input to a single class label. We define state $s = \mathbb{R}^n$ to be the input and $\mathcal{A}$ be the finite set of action labels. Formally, the classification problem can be stated as follows: given training data $(s_1, a_1), ..., (s_k, a_k)$, where $a_i \in \mathcal{A}$, produce a classifier $h : \mathcal{S} \rightarrow \mathcal{A}$ which maps an input s to label $a \in \mathcal{A}$.

Although they map each state to only one action, single-label classification methods can be used to represent equivalent action choices if this choice itself is modeled as its own label. For example, the action *Random* can be used to represent the random selection among all available actions. However, the appropriate use of *Random* must then be demonstrated by the teacher in order to be incorporated into the robot's policy. While this approach could be used to achieve the desired randomized behavior, it would require the teacher to create a new action. Furthermore, the use of this new action would then have to be demonstrated consistently. If the action is demonstrated inconsistently, or if sensor noise is present, the problem of overlapping demonstrations remains.

Another subset of classification approaches, *multi-label classifiers*, are used to learn mappings between a single input and a *set* of labels [83]. Originally motivated by research in document categorization, multi-label classification can be formalized as follows: given training data $(s_1, A_1), ..., (s_k, A_k)$, where $A_i = 2^{|\mathcal{A}|}$ is a vector of binary values indicating which labels are associated with input s_i, produce a classifier $h : \mathcal{S} \rightarrow 2^{|\mathcal{A}|}$ which associates a given input s with a set of labels.

Multi-label classification can also be used to capture equivalent action choices. However, this approach would require the teacher to provide a list of all valid actions at each demonstration, instead of selecting a single action. The resulting interaction would require greater effort from the teacher, and sensor noise and overlapping demonstrations due to sensor noise would remain.

Our goal is to take advantage of the teacher's natural demonstration style, in which a single action is selected for each demonstration, while enabling the robot to model overlapping demonstrations explicitly as action choices. We define our problem as: given training data $(s_1, a_1), ..., (s_k, a_k)$, produce a classifier $h : \mathcal{S} \rightarrow 2^{|\mathcal{A}|}$ which maps a given input s to a binary vector indicating which labels are associated with input s.

5.2 Option Classes

To address the problem of inconsistent demonstrations stemming from the existence of multiple equivalent actions or sensor noise, we introduce *option classes*, which we define as data classes that represent the choice between two or more actions. In this section we present an algorithm for extracting option classes from the underlying data distribution, enabling action choices to be modeled explicitly by the robot's classification-based policy.

Option classes represent the choice between any number of available actions. For our obstacle avoidance domain example, option classes include choices between two actions, such as *Option-Forward-Left (O-FL)* and *Option-Left-Right (O-LR)*, or three actions, *Option-Forward-Left-Right (O-FLR).* Figure 5.2 presents an example of the option class algorithm applied to the distribution from Figure 5.1(b). Note that the middle region of overlapping points now forms the option class *O-RL*, representing the choice between two valid actions,

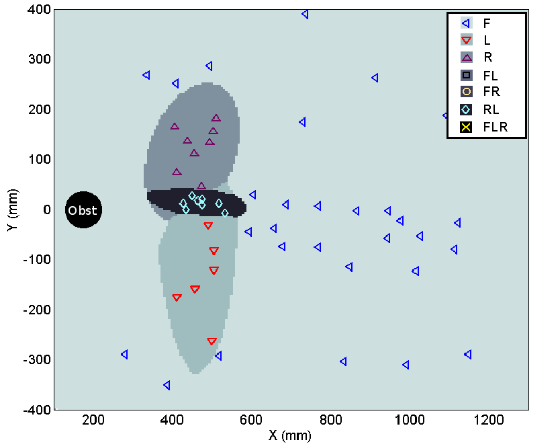


Figure 5.2: Option class policy obtained from the inconsistent demonstration distribution in Figure 5.1(b).

Right and *Left*. The inclusion of this data class results in a complete policy for the entire state space.

The option class algorithm identifies regions of the state space in which data from multiple classes overlaps. It then relables all datapoints within the identified region using a new option class label and retrains the classifier on the resulting dataset.

Algorithm 3 presents the option class algorithm, which is executed each time the classifier is retrained. Given the complete set of demonstrations D, the algorithm identifies the set of datapoints M that fall below the confidence threshold and can not be classified with high confidence (line 3). Additionally, the algorithm calculates the average nearest-neighbor distance over the complete demonstration dataset (line 4). Nearest neighbor distance is defined as the Euclidean distance from the query point to the closest point in the data set. The average nearest neighbor distance over the complete dataset provides the algorithm with a domain-independent method for evaluating point proximity and identifying points that are located close together.

Dataset M and distance d are used to identify clusters of closely located low-confidence points. Searching over set M, the algorithm identifies points that form connected components with a maximum distance d between points (line 5). The resulting set C contains clusters of points that form the candidate option classes. For each connected component, the algorithm calculates the set of actions A represented by the datapoints (line 7).

If the connected component contains more than n points, and consists of two or more action labels, points from the connected component are used to form a new option class representing

Algorithm 3 Option Class Algorithm

```
1: given demonstration dataset $D$
2: UpdateClassifier($D$)
3: $M \leftarrow$ PointsInLowConfidenceRegion($D$)
4: $d \leftarrow$ MeanNearestNeighborDist($D$)
5: $C \leftarrow$ ConnectedComponents($M$,$d$)
6: for $c \in C$ do
7:     $A \leftarrow$ Labels($c$)
8:     if Size($c$) $> n$ and Size($A$) $> 1$ then
9:         CreateLabel($D$, $c$, Option-$A$)
10: UpdateClassifier($D$)
11: ResetLabels($D$)
```

the choice between the actions in A (line 9). Action labels for these points are temporarily changed from their original demonstrated action to *Option-A*. Once all connected components are analyzed, the classifier representing the policy is relearned with the new option class labels. The resulting policy is then saved and used to control robot behavior. To control the robot, all state queries mapped by the policy to an option class are then assigned a single action, which is selected based on the distribution of actions A in the connected component from which the option class was derived.

Once a policy representing the new option classes has been obtained, action labels for all datapoints in dataset D are reset to their original demonstrated action (line 11). The option class acts only as a temporary label and must be re-acquired at the next policy update. We use this mechanism to ensure that any extraneous option classes that form early in the learning process due to insufficient demonstrations are dissolved once additional demonstrations are obtained.

Option classes are an abstract internal representation that enables the robot's policy to explicitly model equivalent action choices. We do not make option classes, such as *Option-Foreward-Left* or *Option-Left-Right*, available to the teacher for demonstration. As a result, the algorithm never acquires training points with option class labels, but must learn them based on the underlying distribution of other action classes.

Additionally, we define special rules for option classes with regards to classification. Based on the assumption that option classes represent a choice between several valid actions, we specify that a datapoint with label x can not be misclassified by an option class representing the choice between actions in A if $x \in A$. For example, points that belong to class F or L and fall into class $O\text{-}FL$ under the new policy are not considered to be misclassified. The opposite case also holds, such that points in O-FL are not misclassified if they fall into F or L. This rule eliminates low confidence regions that may be caused by such misclassifications.

The contributed option class algorithm is classifier independent and contains two parameters, n and d, which are used to control the size and generalization of option classes. The value of n controls the minimum number of points required to create an option class. Smaller values enable the robot to learn action choices from fewer demonstrations, and in our implementation the value of parameter n was set to 3. The variable d represents the average nearest neighbor distance between points, a domain-independent value calculated based on the data distribution of the particular domain. We are able to control the behavior of the algorithm by scaling the value of d. Smaller values require large numbers of demonstrations to form dense clusters of points to create option classes. Larger values lead to greater generalization. Through experimental evaluation we have found that scaling the mean nearest neighbor distance value by 1.5 provides a good balance between generalization and the number of demonstrations.

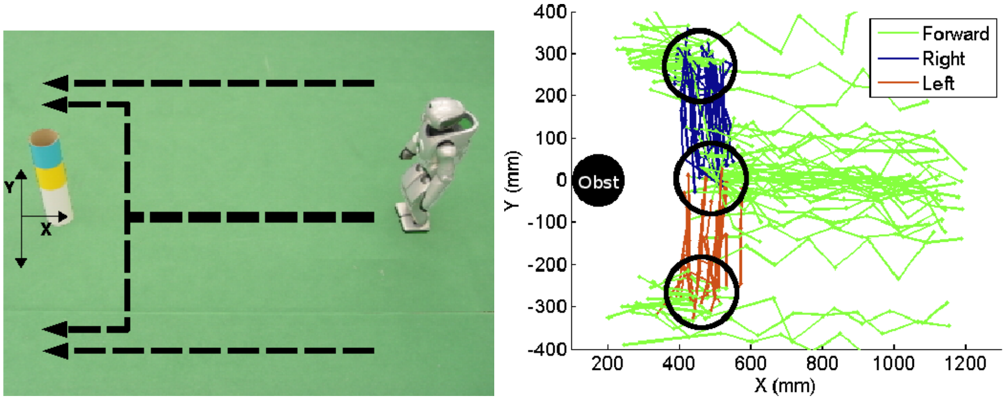
5.3 Experimental Evaluation

In this section, we compare the performance of the Confidence-Based Autonomy algorithm with and without option classes in an obstacle avoidance domain.

5.3.1 Obstacle Avoidance Domain

The obstacle avoidance domain, shown in Figure 5.3(a), requires a Sony QRIO humanoid robot to navigate around a colored post located in its path. The robot has a choice of three actions, $A = \{Forward(F), Left(L), Right(R)\}$, each of which moves the robot approximately 10cm in the designated direction. The robot's state, (x, y), represents its position relative to the post in two-dimensional space as reported by the noisy onboard vision system. We chose a two-dimensional representation for these experiments to enable an intuitive graphical representation of the algorithm's behavior.

Lines in Figure 5.3(a) represent example robot trajectories, with each dash approximately equal to the length of a single robot step. Note that when the robot is positioned directly in front of the post, both avoidance directions, left and right, are equivalently applicable. Once the robot has committed to a direction and has shifted its position, however, it must commit to this course to avoid oscillation. When faced with a choice of directions during



(a) (b)

Figure 5.3: (a) Obstacle avoidance domain with Sony QRIO robot. (b) Traces of 50 demonstration trials used for algorithm evaluation and comparison. Circles highlight areas of inconsistent demonstration.

demonstration, the teacher selects among the valid actions at random.

5.3.2 Evaluation Results

For the evaluation, we recorded 50 complete demonstration trials of the obstacle avoidance task. We used data from the trials, shown in Figure 5.3(b), to provide a consistent library of training points for algorithm comparison. For each algorithm, we performed 20 evaluations by presenting the algorithm with demonstration trials from the library in random order.

We compared the performance of each algorithm with respect to the number of demonstrations and the percentage of complete policies at the end of the training sequence. A count of the number of demonstrations requested by the robot through Confident Execution (CE) and selected by the teacher through Corrective Demonstrations (CD) was also maintained, along with the number of demonstrations falling within the circled option regions (OR) highlighted in Figure 5.3(b). Since multiple valid actions exist for all option regions, classification accuracy is not an informative metric for this experiment.

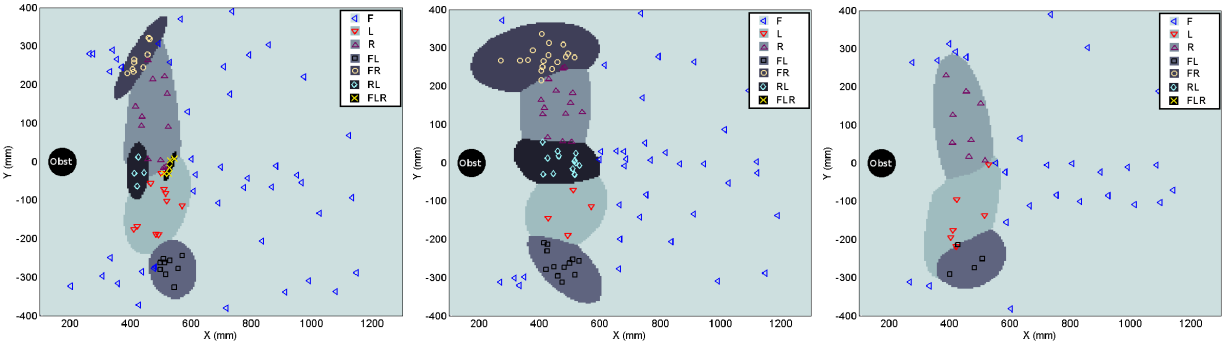
Table 5.1 presents the results of the experiment, which show a significant improvement in learning performance due to option classes. Given the same library of training points, the original Confidence-Based Autonomy algorithm resulted in a complete action policy in only 20% of trials, compared to 90% using the option class method. The standard CBA approach also required a greater number of demonstrations, which can be attributed to

	Complete Policies	Demonstrations			
		Total	CE	CD	OR
CBA	20%	100±41	96±42	4±3	43±19
CBA-Opt	90%	67±12	56±10.6	11±5	15±6

Table 5.1: Comparison of CBA learning performance with and without option classes.

the incomplete policies in which many demonstrations continued to be requested in the persistant low confidence regions. Note that the difference in the number of demonstrations performed within the option regions accounts for most of the difference in the total number of demonstrations between the algorithms.

Figure 5.4 presents three example policies resulting from the experiments. Each policy shows a different, but equally valid combination of option classes. Figure 5.4(a) presents a policy consisting of four option classes. A small region centered near (520,0) represents the option class *O-FRL*, in which any of the three robot actions are valid. As the robot nears the obstacle, the option class becomes *O-RL*, indicating that the Forward action is no longer a valid option. Option classes *O-FR* and *O-FL* represent the other option regions. In Figure 5.4(b) only three option classes form, with a large *O-RL* as the robot approaches the obstacle. In Figure 5.4(c), points within the middle option region are fully separable, resulting in a policy with a slight bias towards passing on the right and only a single option class, *O-FL*. Each of these policies was successfully applied to performing the obstacle avoidance task using the real QRIO robot.



(a) (b) (c)

Figure 5.4: Three example option class policies for the obstacle avoidance domain.

5.4 Chapter Summary

Regions of inconsistent demonstrations are frequently encountered in learning from demonstration due to the existence of multiple applicable actions or sensor noise. In this chapter, we presented a classifier-independent algorithm based on Confidence-Based Autonomy that enables choices between multiple actions to be represented explicitly within the robot's policy through the creation of option classes. Evaluation and comparison of demonstration learning with and without option classes was performed in a real-world, noisy robot domain. Option classes were shown to significantly improve learning performance, enabling the algorithm to converge to a complete policy with far greater frequency and requiring fewer demonstrations.

Chapter 6

Multi-Robot Learning and Collaboration

The previous chapters have considered how a single robot can select and learn from demonstrations. In this chapter, we examine the problem of *multi-robot* learning from demonstration (MLfD), in which a single person teaches multiple robots at the same time.

The problem of multi-robot learning from demonstration has not previously been addressed in robotics research. All demonstration learning approaches developed to date have been designed for single-robot applications, and have relied on close one-to-one interaction between the robot and the teacher. As a result, these techniques do not scale to multi-robot domains due to the problem of *limited human attention* – the fact that the teacher is not able to pay attention to, and interact with, all robots at the same time. While multi-robot behavior may be achieved by these methods through the use of multiple teachers, one for each robot, this approach would be impractical for most real-world applications. To enable a single teacher to work with multiple robots at the same time, a solution is required that endows each robot with a degree of self-regulation and autonomy.

In this thesis, we contribute *flexMLfD*, the first multi-robot demonstration learning framework. Our approach is based on the Confidence-Based Autonomy algorithm presented in the previous chapters. Confidence-Based Autonomy establishes a general state and action representation and provides a means for single-robot policy learning through adjustable autonomy. Using an independent instance of the CBA algorithm, each robot acquires its own set of demonstrations and learns an individual task policy. Specifically, given a group of robots R, each robot $r_j \in R$ uses Confidence-Based Autonomy to learn a policy $\Pi_j : S_j \rightarrow A_j$ mapping from the robot's states to its actions. Each robot may have a unique state and action set, allowing distinct policies to be learned by possibly heterogeneous robots. The general representation and modularity provided by the CBA learning approach and interface result in a flexible task-independent and robot-independent learning framework.

The unique feature of the CBA algorithm that enables it to be applied to multi-robot learning is the Confident Execution component, which enables each robot to regulate its own autonomy, and to pause execution when faced with an uncertain situation. The adjustable autonomy of each robot enables the teacher to switch attention between robots on an as-needed basis.

Algorithm 4 outlines the general procedure followed by the teacher in performing multi-robot demonstrations. The teacher alternates between responding to demonstration requests when they are present, and correcting any mistakes in the autonomous behavior of the robots. The teacher interacts with only a single robot at any one time, while also monitoring the other robots in the background. The function $f(D)$, represents a demonstration request selection policy, such as first-in-first-out or round-robin ordering. In the experiments presented in this thesis, $f(D)$ represents random selection.

Algorithm 4 Multi-robot demonstration

```
Let $D$ be set of current demonstration requests
while the robots request demonstrations or incorrect behavior is observed do
  if $D \neq \emptyset$ then
    - Select robot demonstration request $r_j$ according to function $f(D)$
    - Perform demonstration for robot $r_j$
  else
    - Observe autonomous execution of the robots
    if correction is required for robot $r_j$ then
      - Perform correction for robot $r_j$
```

6.1 Teaching Multi-Robot Collaboration

In addition to teaching multiple robots at the same time, we are interested in teaching them to work together. Solutions to many complex tasks require the collaboration of multiple robots. In this thesis, we study multi-robot learning from demonstration in the context of *loosely-coordinated* tasks, which we define as tasks that contain elements that can be independently performed by individual robots, but that require a degree of coordination to couple their execution.

Our approach to teaching collaborative behavior through demonstration relies on *emergent multi-robot coordination* [64], in which the solution to the shared multi-robot task emerges from the complementary actions performed by robots based on their independent policies. To achieve this coordination, each robot's action abilities may include communication. We define each robot's action set by $A = A_p \cup A_c$, where A_p is the set of physical robot actions and A_c is the set of communication actions. All actions within set A are available to the teacher for demonstration.

Multi-robot coordination requires a rich state representation consisting of both local and communicated information. To this end, we categorize the robot's state features based on their source and purpose; for example, information locally observed by the robot's sensors may be private to the robot (e.g. current wheel angle), shared with its teammates at all times (e.g. robot position), or shared only under particular conditions (e.g. robot position, but only when it is known with high confidence). Specifically, we define the robot's state as $S = \{F_o \cup F_s \cup F_c \cup F_i \cup F_t\}$ where

- F_o = private, locally observed state features
- F_s = locally observed state features that are automatically communicated to teammates each time their value changes
- F_c = locally observed features communicated using communication actions A_c as defined by policy Π
- F_i = internal state features dependent upon other features or robot actions
- F_t = state features containing data either directly contained in, or calculated based on, information communicated from teammates

In this representation, local and communicated data is combined within the robot's state. Coordination between robots occurs when complementary actions are selected by the policy based on this input. Using this representation, we present three methods for teaching emergent multi-robot coordination using demonstration.

6.1.1 Implicit Coordination without Communication

The most basic level of multi-robot coordination is **implicit coordination**, in which physical actions and observed state allow complementary behaviors to occur without communication or shared intent [47]. Using implicit coordination, robots make decisions based only on locally observed information and are often not conscious of the coordination or even of each other's presence. Teaching implicit coordination through demonstration can therefore be reduced to the problem of teaching multiple robots to perform independent tasks at the same time. Within our framework, implicit coordination is represented by the policy

$$\Pi : \{F_o, \emptyset, \emptyset, \emptyset, \emptyset\} \rightarrow \{A_p, \emptyset\}$$

which maps the robot's locally observed state directly to the physical actions. Coordination occurs through the environmental changes resulting from the executed actions.

6.1.2 Coordination through Active Communication

Domains in which information required for coordination can not be obtained through the robot's own sensors require explicit communication, such as wireless messages. **Coordination through active communication** enables the teacher to use demonstration to explicitly teach *when* communication is required. Based on demonstrations of communication actions A_c obtained from the teacher, communication is incorporated directly into a robot's policy along with the physical actions A_p. This technique enables the teacher to specify the conditions under which communication should take place. The resulting policy is defined by:

$$\Pi : \{F_o, \emptyset, F_c, F_i, F_t\} \rightarrow \{A_p, A_c\}$$

While most physical actions have an observable effect that changes the robot's state (i.e. moving an object changes its location), the immediate effect of communication actions is not observable. To prevent the robot from remaining in the same state following a communication action, we utilize internal state features F_i to represent the last communicated value of each element of F_c ($\forall f \in F_c \rightarrow f \in F_i$). A mismatch between the value of a particular feature in F_i and F_c indicates that the local state no longer matches the teammates' knowledge. All state information received from other robots through communication is stored within the set F_t.

6.1.3 Coordination through Shared State

Robot coordination frequently relies on shared state information that must be maintained up to date at *all* times, not just under specific conditions. For example, a robot performing a navigation task with its teammates may always need to know their locations. **Coordination through shared state** automates the communication process for this common case, enabling robot coordination based on automatically updated state features. Specifically, we define F_s as the set of local features that are automatically communicated to teammates each time their value changes. Coordination through shared state is therefore defined by the policy:

$$\Pi : \{F_o, F_s, \emptyset, \emptyset, F_t\} \rightarrow \{A_p, \emptyset\}$$

Since communication occurs automatically, this approach does not require communication actions to be demonstrated or incorporated into the robot policy. Using this technique, the teacher is able to focus on demonstrating only the physical actions to be performed based on state information shared between robots. Note that this approach assumes that shared features do not change very rapidly; attempting to share a sensor value which changes at a high frequency would quickly cause network congestion.

6.1.4 Discussion

Implicit coordination allows collaborative behaviors to be performed without communication, while active communication and shared state rely on communication to coordinate the robots' actions. The difference between the two communication-based approaches is most significant in domains in which communicated state features take on a *range* of values, and in which such features have importance only over a narrow segment of that range. Such features are commonly encountered in robotic problems. Consider, for example, a robot that only needs to know the location of its teammate if the teammate has located an object of interest. Under all other conditions, the teammate's position, if known, would be ignored. In this scenario, the teammate can choose between two communication strategies: 1) to update its location only when it finds an interesting object, or 2) to communicate its location at all times and rely on its teammate to ignore this information when it is not relevant. Both of the approaches is valid, and preference between them depends on the relative costs of sending communication messages and learning to ignore irrelevant information. This is the tradeoff that is captured by the active communication and shared state techniques.

6.2 Ball Sorting Domain

Evaluation of multi-robot learning and coordination was performed in the ball sorting domain using two Sony QRIO humanoid robots. Figure 6.1 shows the robots operating in the domain, which consists of two sorting stations connected by ramps. Each station has an individual queue of colored balls (red, yellow or blue) that arrive via a sloped ramp for sorting. The robots' task is to sort the balls by color into four bins.

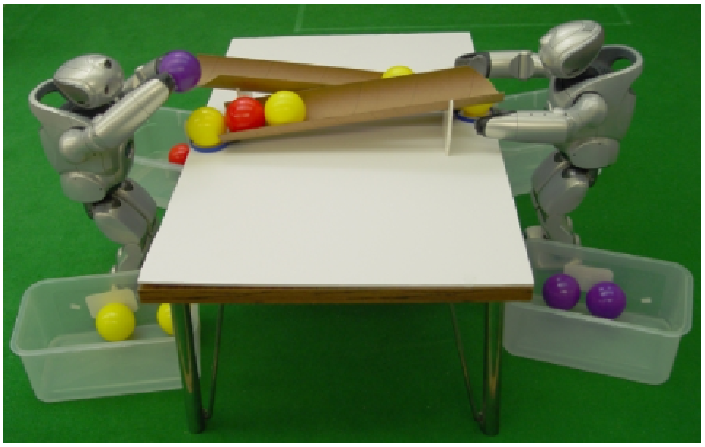


Figure 6.1: QRIO robots performing ball sorting task.

The following set of physical actions is available to each robot: $A_p = \{$*SortLeft, SortRight, PassRamp, Wait, Leave*$\}$. Actions *SortLeft* and *SortRight* enable the robot to pick up a ball and place it into a bin on either side. The *PassRamp* action causes the ball to be placed into the teammate's ramp, where it rolls down and takes position at the tail end of the other robot's queue. The *Wait* and *Leave* actions enable the robot to wait for a short duration or walk away from the table, respectively. The color and location of the balls is determined by each robot using its onboard vision system. Using these abilities, the robot are taught to perform the following three tasks:

Task 1: Each robot begins with multiple balls of various colors in its queue. QRIO A sorts red and yellow balls into the left and right bins, respectively, and passes blue balls to QRIO B. QRIO B sorts blue and yellow balls into the left and right bins, respectively, and passes red balls to QRIO A. If a robot's queue is empty, it should wait until additional balls arrive.

Task 2: Extend Task 1 such that each robot communicates to its teammate the status of its queue, empty or full. When its teammate's queue is empty, a robot in possession of a ball should pass the ball to the teammate's queue. However, only balls that can be sorted by the other robot should be passed. For example, QRIO A should pass only the blue and yellow balls, and QRIO B should pass only the red and yellow balls. If both queues are empty, the robots should wait.

Task 3: Each robot begins with multiple balls of various colors in its queue. QRIO A first sorts all of the red balls on the table into its left bin, while QRIO B passes balls of all colors, thereby helping to rotate the queue. Once all the red balls are sorted, QRIO B sorts all the blue balls into its left bin while QRIO A passes. Once all the blue balls are sorted, both robots sort the remaining yellow balls into their right bins. Whenever both queues are empty, the task is complete and the robots leave the table.

Each of the above tasks is designed to test different aspects of multi-robot teaching and coordination. In the first task, coordination between robots emerges naturally based solely on the physical actions of the robots and communication is not required. Task 2 requires coordination through communication to ensure that the sorted balls are distributed more evenly between the robots, while Task 3 adds ordering constraints and additional coordination requirements.

6.3 Ball Sorting Example

Our evaluation of the learning algorithm and teaching approaches utilizes continuous and noisy features. Before presenting this evaluation, we provide a walk-though of a simplified example of Task 1 with a noiseless, boolean state representation. Only a single demonstration of each state is required in this representation, helping to illustrate elements of the learning process. To achieve this representation, we calibrate the robot's onboard vision system to classify each ball into one of the discrete color classes, such that $F_o = \{red, yellow, blue\}$.

Figure 6.2 presents an interactive learning sequence between the teacher (T) and the robots (R). The robots begin in the top configuration, with no initial knowledge about the task. Both robots request a demonstration upon encountering the first state, and are instructed by the teacher to place their respective balls, red and blue, into the left bin. Upon receiving their individual instructions, each robot executes the specified action.

The second diagram in the figure shows the task state after both robots have completed their first action. One ball has been sorted on each side, and the next ball in the queue is now available to each robot. QRIO B receives a second blue ball, and therefore executes the previously demonstrated action autonomously. QRIO A requests a demonstration request for the new ball color, yellow.

At each following timestep, the robots observe their state, compare it to previously encountered examples, and select between autonomous execution and demonstration. At steps 3 and 5 the robots are taught to pass balls to their teammate, causing these balls to appear at the end of the other robot's queue. The learned policy also allows the robots to wait while no balls are present, and respond once a ball has been received (QRIO A, steps 4-6). Although the figure presents the learning process as a sequence of synchronized steps, no synchronization occurs in real-life learning. The entire task requires 8 demonstrations, 4 per robot, to learn.

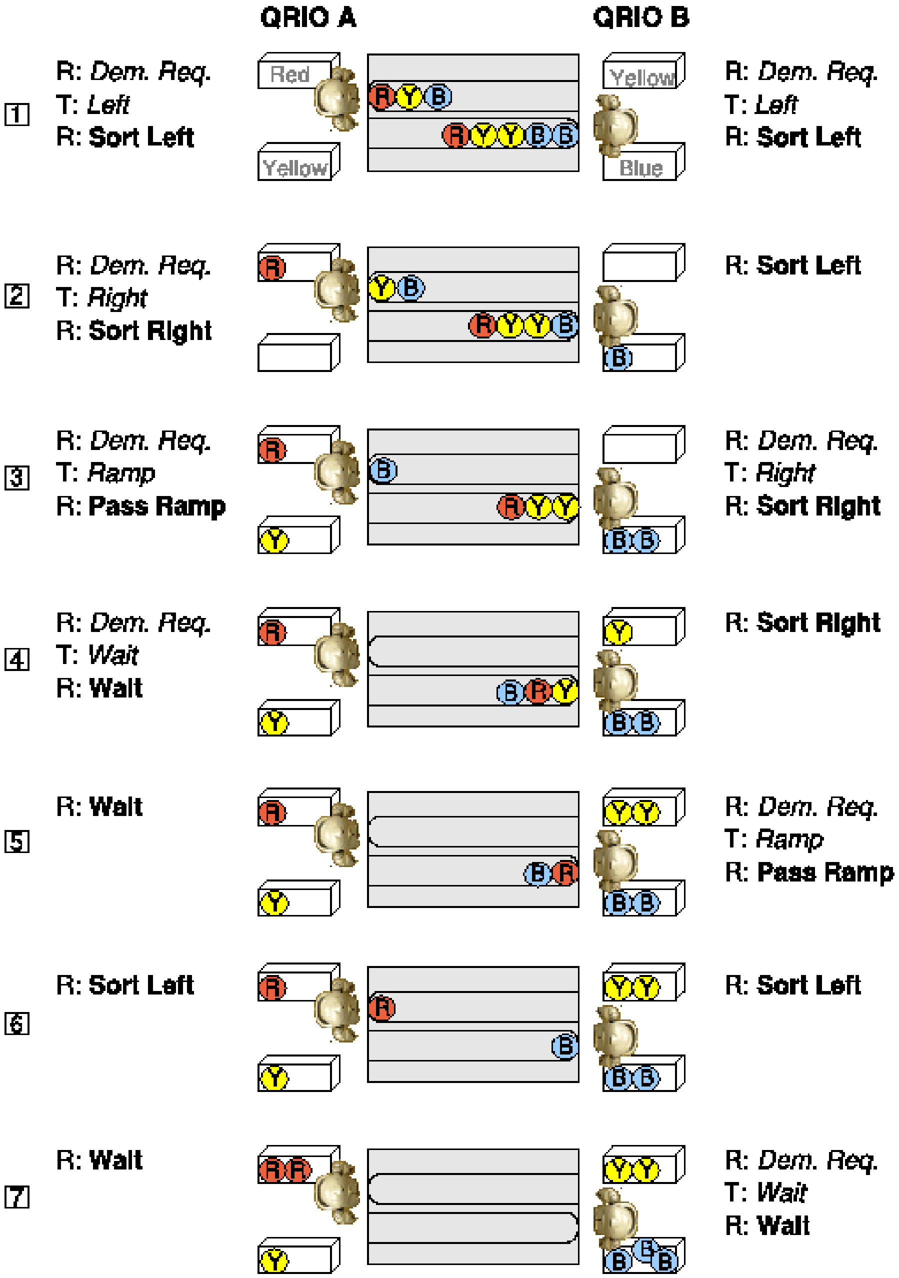

QRIO A
QRIO B
1
R: *Dem. Req.*
T: *Left*
R: **Sort Left**
Red
Yellow
R Y B
R Y Y B B
Yellow
Blue
R: *Dem. Req.*
T: *Left*
R: **Sort Left**
2
R: *Dem. Req.*
T: *Right*
R: **Sort Right**
R: **Sort Left**
3
R: *Dem. Req.*
T: *Ramp*
R: **Pass Ramp**
R: *Dem. Req.*
T: *Right*
R: **Sort Right**
4
R: *Dem. Req.*
T: *Wait*
R: **Wait**
R: **Sort Right**
5
R: **Wait**
R: *Dem. Req.*
T: *Ramp*
R: **Pass Ramp**
6
R: **Sort Left**
R: **Sort Left**
7
R: **Wait**
R: *Dem. Req.*
T: *Wait*
R: **Wait**

Figure 6.2: Collaborative ball sorting example.

6.4 Evaluation Results

In this section, we present an evaluation of our three approaches to teaching multi-robot coordination. We begin by evaluating each coordination approach independently by applying it to one of the ball sorting tasks. We then evaluate the performance of the communication-based coordination approaches in greater detail using Task 3.

For all evaluations, the robots observe ball color as the average RGB values of the pixels in the detected ball region. Results are reported as the total number of demonstrations required to learn the task, averaged over ten trials. Note that in this discussion we do not distinguishing between demonstrations obtained from Confident Execution and Corrective Demonstration methods. While most demonstrations are initiated by the robot through Confident Execution, the Corrective Demonstration algorithm does play a significant role in aiding policy learning of each individual robot. Consider, for example, a robot that first observes demonstrations of passing both a blue and a yellow ball to a teammate. Upon encountering a red ball, the robot may assume, due to generalization of the classifier, that the same action should also be performed for this new color. Corrective Demonstration enables the teacher to correct this assumption if necessary.

6.4.1 Implicit Coordination without Communication

We evaluate implicit coordination learning by training Task 1, in which the teacher's demonstrations are limited to the robot's physical actions A_p (note that the *Leave* action is not used for tasks 1 and 2). The following state representation, consisting only of locally observed information, is used:

- $F_o = \{R, G, B\}$
- $F_s = F_c = F_i = F_t = \emptyset$

Using this representation, both robots successfully learned their individual policies, which enabled them to collaboratively sort the balls by coordinating through their actions. Training the entire task required an average of 20 demonstrations (10 per robot), with a standard deviation of 1.1.

Note that during the execution of the task, the robots encounter both noisy and noiseless states. The number of demonstrations required to learn each state-action mapping is proportional to the level of noise in the sensor readings. For example, an empty queue contains no ball color information and consistently results in the state vector S ={0,0,0}. Since this value is not affected by noise, only a single demonstration is required to teach each robot to *Wait* when the queue is empty. Ball color readings, on the other hand, vary due to both sensor noise and slight variations in the robot's view angle and ball distance between actions. An average of 3.1 demonstrations are therefore required for the model to learn to generalize over each ball color class.

6.4.2 Coordination through Active Communication

Coordination through active communication enables communication to be represented directly within each robot's policy. In this section, we evaluate this approach using Task 2 and the following state representation:

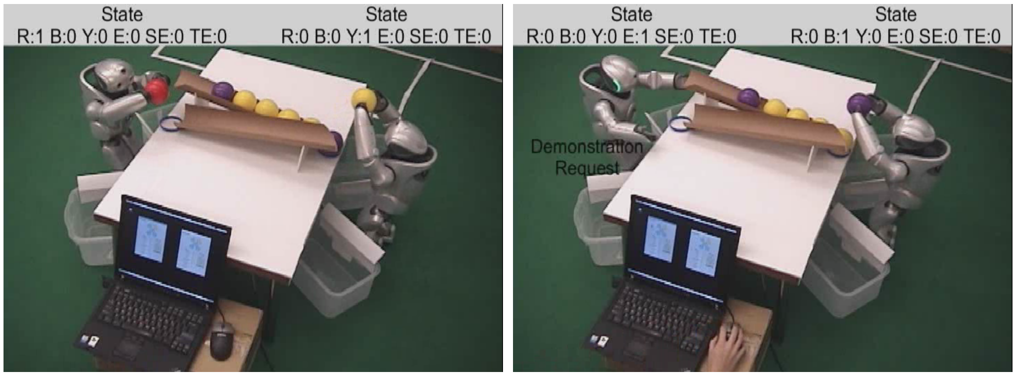
- $F_o = \{R, G, B\}$
- $F_s = \emptyset$
- $F_c = F_i = F_t = \{Empty\}$

We define a new boolean feature, F_c ={*Empty*}, to represent the status of the robot's ball queue, empty or full. This feature is locally observed and communicated to each robot's teammate using communication action $A_c = \{SendEmpty\}$. For each robot, feature sets F_i and F_t contain the robot's own last communicated value of *Empty* and the most recent value of *Empty* received from its teammate, respectively.

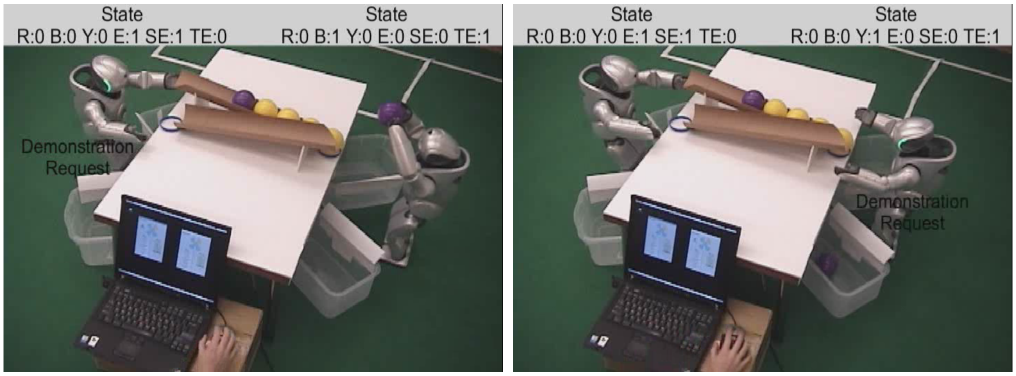
The complete state vector consists of six features. Note that all of the information available to a robot, whether continuous or discrete, locally observed or communicated, is combined to form the robot's state vector, allowing the algorithm to generalize over all of these variations. Each feature value is typically normalized by the classifier prior to analysis to give each feature equal weight.

Using the above setup, the teacher required an average of 35 demonstrations to teach Task 2. While each ball color still required multiple demonstrations, the algorithm was able to rapidly generalize over the boolean *Empty* features. For example, learning that communication updates must be performed each time the value of the feature *Empty* in F_c and F_i does not match, regardless of the ball color being observed, required only two demonstrations.

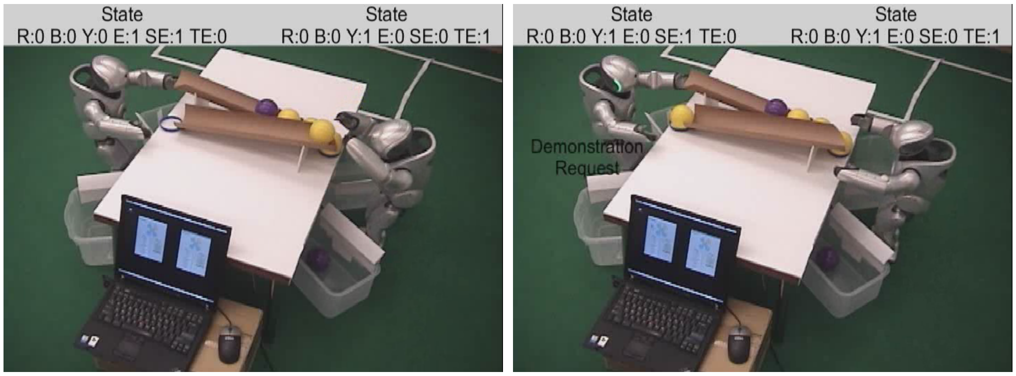
Figure 6.3 presents a sequence of images showing how the teacher uses demonstration to teach the ball sorting task using active communication. The teacher uses the laptop computer


State
R:1 B:0 Y:0 E:0 SE:0 TE:0
State
R:0 B:0 Y:1 E:0 SE:0 TE:0
State
R:0 B:0 Y:0 E:1 SE:0 TE:0
State
R:0 B:1 Y:0 E:0 SE:0 TE:0
Demonstration
Request

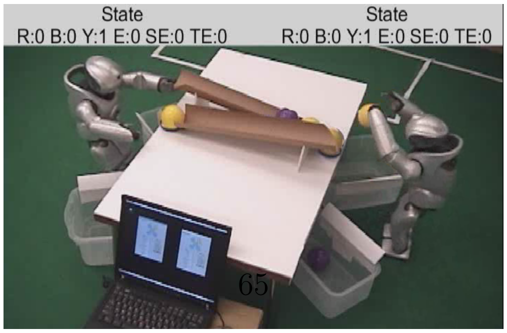
(a) {SortLeft, SortRight} (b) {DemReq, SortLeft}


State
R:0 B:0 Y:0 E:1 SE:1 TE:0
State
R:0 B:1 Y:0 E:0 SE:0 TE:1
Demonstration
Request
State
R:0 B:0 Y:0 E:1 SE:1 TE:0
State
R:0 B:0 Y:1 E:0 SE:0 TE:1
Demonstration
Request

(c) {DemReq, SortLeft} (d) {Wait, DemReq}


State
R:0 B:0 Y:0 E:1 SE:1 TE:0
State
R:0 B:0 Y:1 E:0 SE:0 TE:1
State
R:0 B:0 Y:1 E:0 SE:1 TE:0
State
R:0 B:0 Y:1 E:0 SE:0 TE:1
Demonstration
Request

(e) {Wait, Pass} (f) {DemReq, SortRight}


State
R:0 B:0 Y:1 E:0 SE:0 TE:0
State
R:0 B:0 Y:1 E:0 SE:0 TE:0

(g) {SortRight, SortRight}

Figure 6.3: Example Task 2 learning sequence using coordination through active communication.

shown at the bottom of the image to perform demonstrations independently for each robot using its own instance of the CBA GUI interface. Each figure is annotated with the robot's current state, shown at the top of the image.

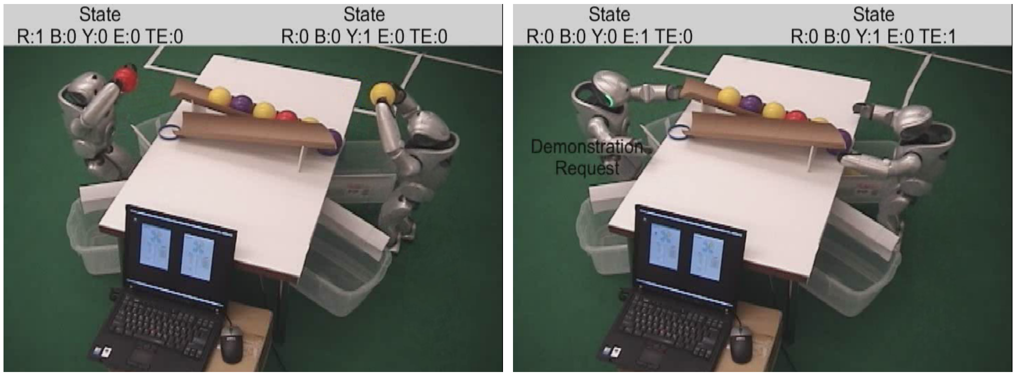
In Figure 6.3(a), the QRIO on the left sorts a red ball into its left bin, while the QRIO on the right sorts its yellow ball into its right bin. After completing its action, the left QRIO observes that its ramp is empty and stops to request a demonstration from the teacher. During this time, the right QRIO continues to perform its task, sorting the next ball in its queue, which is blue, into its left bin. Note that since communication between robots has not yet occurred, the state of the right QRIO does not accurately reflect its teammate's status because the teammate's empty (TE) state feature value is 0.

In response to the demonstration request, the teacher instructs the left QRIO to communicate its queue status using the *SendEmpty* action. In Figure 6.3(c) we observe that communication has occurred and the right robot is now aware that its teammate's queue is empty. The left QRIO requests a demonstration for its new state, and the teacher instructs it to $Wait$. In Figure 6.3(d) the right QRIO requests a demonstration for what to do given that it has a yellow ball and its teammate's queue is empty. The teacher instructs the robot to pass this ball to its teammate; the robot is shown performing this action in Figure 6.3(e). Once the left QRIO observes that a yellow ball has arrived, it requests a demonstration and the teacher instructs the robot to communicate its updated ramp status to its teammate to indicate that the queue is no longer empty. Once this information is updated, the left QRIO continues by sorting the yellow ball into its right bin. During this time the right QRIO autonomously continues with the ball sorting task, sorting its next yellow ball into the right bin. Once the left robot's queue becomes empty again, it will autonomously communicate the status of its queue to the other robot, and the ball sharing process will repeat autonomously.

6.4.3 Coordination through Shared State

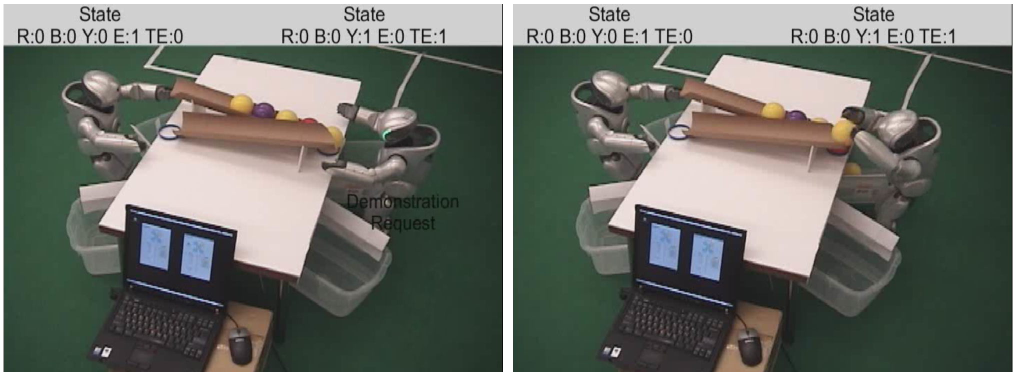
Coordination through shared state automates the communication process, leaving the teacher to demonstrate the robot's physical behavior based on shared information. In the place of explicit communication actions, this technique utilizes a set of *shared state features*. Any of the robot's locally observed state features may be selected by the teacher to be shared with a teammate. The status of each shared feature is then tracked by the system, and updates are communicated to the teammate each time the value changes.

Coordination through shared state was similarly evaluated using Task 2. For this task, state information shared between robots consists of the state feature *Empty*. The complete


State
R:1 B:0 Y:0 E:0 TE:0
State
R:0 B:0 Y:1 E:0 TE:0
State
R:0 B:0 Y:0 E:1 TE:0
State
R:0 B:0 Y:1 E:0 TE:1
Demonstration Request

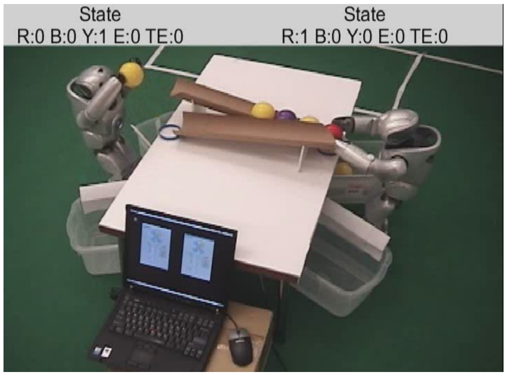
(a) {SortLeft, SortRight}

(b) {DemReq, SortLeft}


State
R:0 B:0 Y:0 E:1 TE:0
State
R:0 B:0 Y:1 E:0 TE:1
Demonstration Request
State
R:0 B:0 Y:0 E:1 TE:0
State
R:0 B:0 Y:1 E:0 TE:1

(c) {Wait, DemReq}

(d) {Wait, Pass}


State
R:0 B:0 Y:1 E:0 TE:0
State
R:1 B:0 Y:0 E:0 TE:0

(e) {SortRight, Pass}

Figure 6.4: Example Task 2 learning sequence using coordination through shared state.

complete state is represented by:

- $F_o = \{R, G, B\}$
- $F_s = F_t = \{Empty\}$
- $F_c = F_i = \emptyset$

where F_s ={*Empty*} represents a robot's local shared ramp status, and F_t ={*Empty*} represents the ramp status of its teammate. The complete state contains five features, and the entire task required an average of 27 demonstrations to learn.

Figure 6.4 presents a sequence of images that shows the robots learning to perform Task 2 using collaboration through shared state. Figure 6.4(a) begins with both robots performing autonomous sorting, with the left robot sorting its last ball into its left bin. Figure 6.4(b) shows what happens once the left QRIO observes that its queue is now empty. Unlike the active communication approach described in the previous section, the value of the robot's *Empty* feature is automatically communicated to its teammate, such that the teammate's empty (TE) state feature for the right QRIO is set to 1. The left QRIO requests a demonstration, asking for instructions about what to do when its queue is empty. The teacher instructs the robot to $Wait$. In Figure 6.4(c), in response to a demonstration request the teacher instructs the right QRIO to pass its yellow ball to its teammate, as shown in Figure 6.4(d). Once the ball arrives at the bottom of the ramp, the left QRIO automatically communicates its updated ramp status to its teammate and autonomously resumes the sorting process, Figure 6.4(e).

6.4.4 Combined Approach and Comparison

In the above evaluation, we showed that each of the presented coordination approaches can be successfully used to learn a variation of the ball sorting task. In this section, we further compare and evaluate our two communication-based methods – coordination through shared state and coordination through active communication – as well as a combined approach utilizing a combination of these techniques.

While our evaluation of Task 2 shows the feasibility of both active communication and shared state, it provides little information about the tradeoffs between the two approaches. The communication requirements of Task 2 are very limited, with only a single communicated boolean feature that had to be updated each time its value changed. A more informative analysis can be obtained by examining a domain with communicated state features that take on a *range* of values, and in which such features have importance only over a narrow segment of that range.

In this section, we use Task 3 to compare the performance of the three communication-based coordination approaches with respect to the number of demonstrations and number of communication messages. Task 3 is a variation of the ball sorting task, in which the robots must sort balls in the order of their color class, such that all red balls are sorted into their bin first, then blue, and finally yellow. Once sorting is complete, the robots turn around and walk away from their sorting stations. To represent this domain, we add two new types of state features. The feature $SortColor$ is used to represent the current color being sorted (red, blue or yellow), and the feature $PassCount$ is used to represent the number of consecutive $PassRamp$ actions that have been executed by a particular robot. Since the robots do not have a global view of the world, the $PassCount$ feature is required to determine when all balls of a particular color have been removed. For example, when sorting red balls, both robots pass any blue or yellow balls they encounter into their teammate's ramp. The resulting effect is that the entire queue of balls rotates, enabling the robots to examine all balls one at a time. Once the value of the $PassCount$ feature for both robots passes some threshold, in our case 10, this indicates that the entire queue has been examined and no additional balls of the current $SortColor$ remain on the table.

In total, three pieces of information are communicated between robots: 1) the current color being sorted ($SortColor$), 2) the number of consecutive passes each robot has performed ($PassCount$), and 3) the status of each robot's ball queue (*Empty*). Actions available to the robots for performing this task include the previously defined physical actions A_p, and the communication actions *SendEmpty*, *SendPassCount*, and $IncrementSortColor$.

Before discussing each coordination approach, we define the $SortColor$ state feature in detail. Unlike previously encountered communicated features, the sorting color is a value that is common to both robots; to achieve accurate performance, the robots must maintain the same value for this feature at all times. As a result, instead of representing this value both as a local and teammate copy of the information, we use a single value for each robot. The value of this feature can be updated both locally and through communication from the teammate using the $IncrementSortColor$ action. Specifically, the execution of $IncrementSortColor$ by a robot first increments its local copy of the variable, and then communicates the new value to its teammate where it immediately updates the other robot's state. Additionally, this action resets the value of the $PassCount$ feature to 0. In summary, this approach achieves state synchrony between robots through the use of a single feature and a multi-function communication action.

	Active Communication	**Shared State**	**Combined**
F_o	$\{R, G, B\}$	$\{R, G, B\}$	$\{R, G, B\}$
F_s	$\emptyset$	$\{PassCount, Empty\}$	$\{Empty\}$
F_c	$\{PassCount, Empty\}$	$\emptyset$	$\{PassCount\}$
F_i	$\{PassCount, Empty\}$	$\emptyset$	$\{PassCount\}$
F_t	$\{PassCount, Empty, SortColor\}$	$\{PassCount, Empty, SortColor\}$	$\{PassCount, Empty, SortColor\}$
A_c	$\{IncSortColor$ $SendPassCount, SendEmpty\}$	$\{IncSortColor\}$	$\{IncSortColor, SendPassCount\}$

Table 6.1: Representation of Task 3 using active communication, shared state and a combined approach.

The above representation for the *SortColor* feature is used for each coordination approach applied to Task 3. Table 6.1 presents a summary of the complete state representations used by each learning method for this task. Note that for each approach, the information locally observed by each robot (F_o) and received from its teammate (F_t) remains the same. The differences consist in the local representation of the state features to be communicated. Below we summarize the distinguishing features of each approach.

Active Communication – In the active communication approach, both the *PassCount* and *Empty* features are communicated explicitly using communication actions. The teacher selects the $PassCount$ feature to be communicated each time its value reaches 10. This representation requires a total of 10 state features and 8 action classes.

Shared State – In the shared state approach, the values of all communicated features are shared each time their value changes. The main difference between this approach and the active communication method is that each robot always knows the exact number of passes that its teammate has performed. This approach utilizes a more compact representation, requiring 8 state features and 6 action classes.

Combined Approach – In the combined approach, active communication is used for some state features, and shared state for others. Specifically, shared state is used for the *Empty* feature, the value of which must be communicated each time it changes. Active communication is used for the *PassCount* feature, the value of which is communicated only once it passes a set threshold[1]. This representation uses a total of 9 state features and 7 action classes.

Table 6.2 presents the results of this experiment in terms of the number of demonstrations required to learn the task policy, and the average number of communication messages sent by each robot during the execution of this policy. We find that the shared state approach requires significantly fewer demonstrations than both active communication and the combined approach. Even in the presence of variables spanning a range of values that contain no useful information, the shared state approach requires less training data. This suggests that learning a threshold for a particular input feature (in our case *PassCount*) is easier for the underlying classifier than dealing with an additional state feature ($F_i = PassCount$) and class label ($SendPassCount$).

The number of communication messages required to perform the task is dependent upon the number, order and color of balls to be sorted. Table 6.2 presents the average number of communication messages required to sort 10 balls of random color and sequence. Analysis of the average number of communication messages reveals the tradeoff between the coordination approaches. The active communication approach utilizes significantly fewer communication messages than shared state since it does not communicate the value of *PassCount* each time it changes.

Additionally, this evaluation highlights the benefit of using a combined approach that takes advantage of both training methods. The combined approach provides the communication benefits of active communication by communicating *PassCount* only when necessary, while

	Number of Demonstrations	**Number of Communication Messages**
Active Communication	135.0 ± 5.9	5.8 ± 1.1
Shared State	52.6 ± 6.6	37.7 ± 2.6
Combined	92.2 ± 4.8	5.8 ± 1.1

Table 6.2: Task 3 evaluation result summary. Comparison of the average number of demonstrations required per robot to learn the task policy, and the average number of communication messages sent by each robot while applying the final policy to sorting 10 randomly colored balls.

---

[1] Alternatively, *PassCount* could also be defined as a boolean feature representing whether enough passes have occurred or not. We do not use this representation here for evaluation purposes.

also reducing the number of demonstrations by using state sharing for features that must be communicated each time they change. The result is a low communication rate and demonstration requirements in between those of either method alone.

In summary, we find that complex domains are likely to benefit from a combination of active communication and shared state techniques if a cost is associated with communication. If communication is free, the shared state approach should be used since it results in the fewest number of demonstrations.

6.5 Heterogeneous Robot Teams

The previous sections presented evaluation of multi-robot learning and coordination using two identical humanoid robots. However, many robotic tasks benefit from combining heterogeneous robots with different abilities. In this section, we highlight the generality of our system and present the playground domain, which showcases collaborative behavior between two QRIOs and four AIBOs.

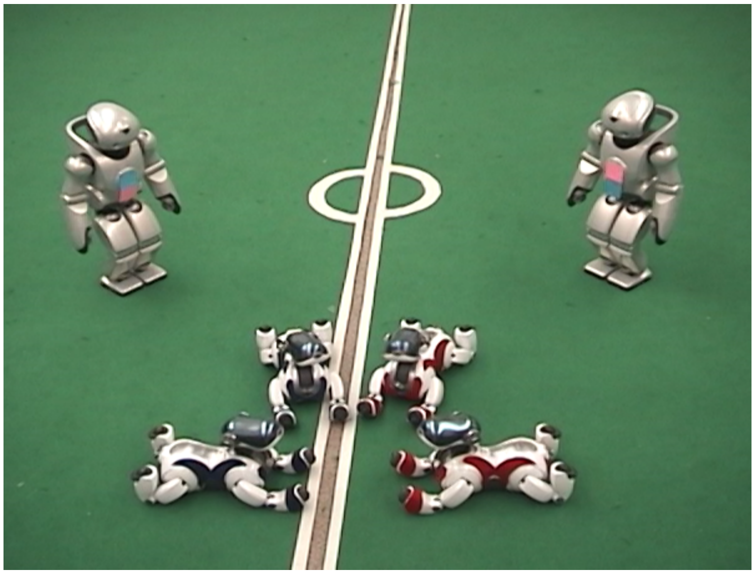


Figure 6.5: Playground domain.

6.5.1 Playground Domain

The playground domain, shown in Figure 6.5, consists of an open space simulating a school playground. Two humanoid QRIO robots represent "teachers", and four AIBO robots represent "students"; each QRIO is assigned two AIBO students as its "class". The task simulates a playground scenario in in which the teachers collect their students at the end of recess and take them back to lessons. The task begins with recess, during which the QRIO robots talk to each other while the AIBOs play. Once the bell sounds, the QRIOs stop their conversation and walk to open spaces on opposite sides of the playground. Each QRIO then calls its respective class and waits for it to arrive. The AIBOs play in the open space until they are called. Once called, each robot navigates to its QRIO teacher. Once a QRIO has all of its students around it, it leaves the playground with the AIBOs following.

In order to enable the AIBOs to visually distinguish the QRIO robots, we marked each humanoid with a two-colored marker on its chest and back. The orientation and size of the marker enabled the AIBOs to identify each robot and determine its relative distance. Blue and red stickers on the AIBOs enable the teacher to distinguish robots in each class. However, the body color of each AIBO was not encoded into the domain representation and was not known to the robots.

We use the following state and action representation for the AIBO robots:

- $A = \{Forward, Left, Right, Stop, Search, Play\}$
- $F_o = \{Q1_a, Q1_d, Q2_a, Q2_d\}$
- $F_s = \{Q1_{near}, Q2_{near}\}$
- $F_t = \{TeacherCall\}$
- $F_c = F_i = \emptyset$

The action set of each robot consists of low-level navigation commands, as well as the ability to search for the teacher and to "play" – a behavior that consists of the robots in a relaxed position on the ground as shown in Figure 6.5. State information consists of the robot's relative distance (Q_d) and angle (Q_a) to each QRIO. An additional boolean feature, Q_{near}, indicates that the AIBO is located in close proximity to its teacher. The value of this feature is shared between the AIBOs and QRIOs, enabling the teacher to know when its students are ready to leave the area. When the position of a QRIO is unknown, we set Q_{near} to 0 and the distance and angle to the default values of 4000 mm and 1.8 rad, respectively, to indicate that the robot is far away. Finally, the AIBOs in each class also receive their teacher's call over the wireless network.

We use the following state and action representation for the QRIO robots:

- $A = \{Talk, WalkToOpenSpace,$ *CallAIBOs, Wait*, *LeadAIBOs*$\}$
- $F_o = \{Bell, InOpenSpace,$ *CalledAIBOs*$\}$
- $F_t = NumStudents$
- $F_c = F_i = F_s = \emptyset$

The action set consists of high-level behaviors, such as navigation to areas of the state space, and the action $Talk$, which causes the QRIO to use conversational pose and gestures to simulate speaking. The observed features of each robot consist of a mixture of sensory information (audible bell sound and location in space), and a history of its past activities (whether the AIBOs have been called). The QRIOs do not share their own state information with each other or the AIBOs. They do, however, listen to AIBO messages indicating their proximity, enabling each robot to know how many students have reached it ($NumStudents$).

In summary, coordination between robots is achieved using a combination of active communication and shared state techniques. Active communication is used by the QRIOs to call the AIBOs, while the AIBOs use shared state to notify the QRIOs when they are near their respective teacher and ready to leave. Additionally, we note that the goal behavior for both AIBOs in each class is the same – both robots have to stop playing at the same time, and then locate, reach and follow the same teacher. As a result, we used common policy learning (see Section 7.2.3) to learn a single policy for these robots using shared demonstrations.

Using these teaching techniques, training the complete task required 6 demonstrations for each QRIO, and an average of 16 demonstrations for each AIBO. A greater number of demonstrations were required for the AIBOs due to their noisy, real-valued state representation.

6.6 Chapter Summary

This chapter introduced *flexMLfD*, a robot-independent and task-independent multi-robot demonstration learning framework based on the Confidence-Based Autonomy algorithm. Using this framework, we contributed three approaches for teaching collaborative multi-robot behavior based on different information sharing strategies. Evaluation of collaborative task learning was performed in two multi-robot domains – ball sorting and playground. Differences between these domains showcase the flexibility of the proposed system, including the following features:

- Multiple robotic platforms, the Sony QRIO and AIBO.
- Variable number of robots, from 2 to 6.
- Heterogeneous and homogeneous groups of robots.
- Flexible action representation supports wide range of actions, including low-level navigation commands and high-level behaviors.
- Flexible state representation includes boolean, discrete and real-valued features and both locally observed and communicated information.
- A combination of active communication and shared state techniques is used to control the frequency with which robots share information.
- Policy learning supports variable degrees of collaboration and interaction between robots, ranging from physically separate but collaborating QRIO robots in the ball sorting task, to physically interacting but independent AIBO robots in the playground domain.

All of the above variations are fully supported by the *flexMLfD* learning system and no special adaptations to the underlying architecture are required for each task. Instead all task-specific elements are specified within the system XML configuration file prior to training.

Chapter 7

Scalability Analysis

In the previous chapter, we introduced multi-robot learning from demonstration and presented three ways in which a human teacher can teach robots to coordinate their actions. Controlling and teaching multiple robots at the same time is a challenging problem. In this chapter, we present a case study analysis the scalability of the *flexMLfD* framework. Our analysis uses a multi-robot domain in which communication and coordination between robots is achieved through shared state. Through experimental evaluation, we examine how the number of robots being taught by the teacher affects the number of demonstrations required to learn the task, the time and attention demands on the teacher, and the delay each robot experiences in obtaining a demonstration.

Using up to seven Sony AIBO robots, we evaluate three approaches to teaching multiple robots at the same time:

- **Synchronous Learning Start Times** – Each robot learns an individual task policy. All robots begin learning at the same time.
- **Offset Learning Start Times** – Each robot learns an individual task policy. Learning begins with a single robot. All remaining robots are introduced incrementally, one at a time, once all previous robots have gained partial autonomy at the task.
- **Common Policy Learning** – All robots begin learning at the same time and learn a single common policy by consolidating their knowledge and sharing demonstration data.

In each approach, robots begin with no knowledge about the task, and learning is complete once all robots are able to perform the task correctly. The synchronous and offset learning approaches examine the most general learning scenario, one in which each robot learns an individual policy, allowing different tasks or roles to be taught at the same time. In the synchronous learning approach, all robots begin at the same time. As we show in our evaluation, this approach places great demand on the teacher for demonstrations early in the training process. The offset learning approach presents an alternate method in which robots are introduced one at a time, after all active learners have already gained partial autonomy at the task, thereby helping to disperse the demand for demonstrations over a longer time period. Common policy learning examines a special case of demonstration learning in which robots consolidate all their demonstration knowledge to learn a single common policy. We evaluate the scalability of our multi-robot learning framework in a beacon homing domain using Sony AIBO robots.

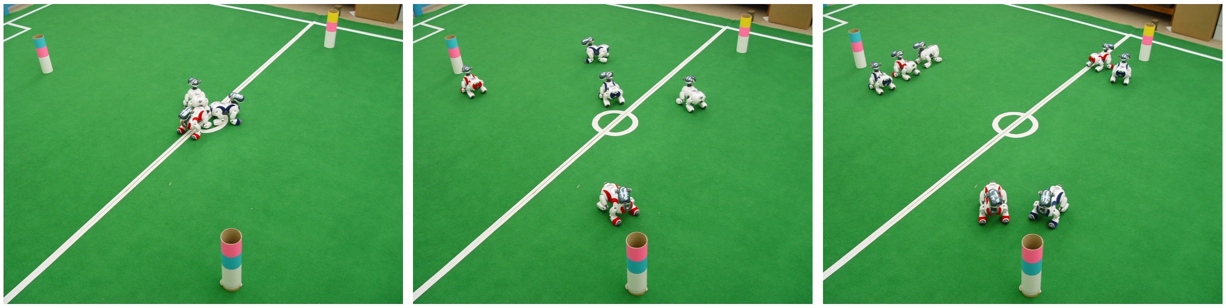
7.1 Beacon Homing Domain

Figure 7.1 shows three examples of AIBO robots operating in the beacon homing domain, which consists of an open area with three uniquely-colored beacons ($B = \{B_1, B_2, B_3\}$) located around the perimeter. Each robot is able to observe the relative position of a beacon using its onboard camera, and to communicate information via the wireless network. The set of action available to each robot is limited to basic movement commands, $A = \{Forward, Left, Right, Search, Stop\}$, used by each robot to navigate in the environment.

Using the representation defined in Section 6.1, we define the robot's state as $s = \{F_o \cup F_s \cup F_c \cup F_i \cup F_t\}$, where:

- $F_o = \{d_1, d_2, d_3, a_1, a_2, a_3\}$
- $F_s = \{myBeacon\}$
- $F_t = \{n_1, n_2, n_3\}$
- $F_c = F_i = \emptyset$

The set of observed features, F_o, contains information about the robot's relative distance (d_i) and angle (a_i) to each beacon $i \in B$. For any beacon not currently in view, the distance and angle are set to the default values 4000 mm and 1.8 rad, respectively, to indicate that this beacon is far away. The set of shared state features, F_s, contains a single value, $myBeacon$, which is set to a beacon's ID number if the robot is within a set distance x of a beacon,



(a) (b) (c)

Figure 7.1: Beacon homing domain: (a) Example starting configuration, 3 robots. (b) Example intermediate stage, 5 robots. (c) Example final configuration, 7 robots.

and -1 if the robot is not located near a beacon. Each robot communicates the value of this feature to its teammates. In turn, all robots use this shared information to determine the values of state features $F_t = \{n_1, n_2, n_3\}$, which maintain the count of the current number of robots occupying each beacon.

In summary, using the above representation, each robot knows its position relative to beacons that it observes, and the number of other robots already located at each of the beacons. Using this information, the teacher can teach each robot to navigate from a random initial location in the center of the open region to one of the colored beacons. Specifically, the selection of a beacon is governed by the following rules: *Given a maximum limit m for the number of robots that can occupy a marker, search until beacon i is found for which the number of robots, n_i, is less than m. Navigate to that beacon and occupy it by stopping within a set distance x. If at any point the number of robots at the selected beacon exceeds m, search for another beacon.*

These explicit rules of the task are known only to the teacher. During the learning process, each robot in the experiment learns an independent policy representing this behavior from demonstrations. All robots were taught the same task to ensure a fair comparison between robots for the scalability evaluation. We set the maximum number of robots allowed per beacon for each experiment to $m = \lceil \frac{\#Robots}{\#Beacons} \rceil$, such that at least one beacon must contain the maximum number of robots. Each experiment began with all robots located in the center of the open region (Figure 7.1(a)) and ended once all robots had reached a beacon (Figure 7.1(c)). Training continued until all robots executed the desired behavior efficiently and correctly without requesting demonstrations.

7.2 Evaluation

The scalability of the *flexMLfD* approach was evaluated in the beacon homing domain using 1, 3, 5, and 7 robots. In this section, we discuss how the number of robots taught by the teacher at the same time affects the number of demonstrations required to learn the task, the demands for time and attention placed on the teacher, and the delay that each robot experiences in obtaining a demonstration.

All evaluations were performed with a single teacher. As with all human user trials, we must account for the fact that the human teacher also learns and adapts over the course of the evaluation. To counter this effect, the teacher performed a practice run of each experiment, which we then discarded from the evaluation. An alternate evaluation method would be to eliminate the human factor by using a standard controller to respond to all demonstration requests in a consistent manner. This approach, however, would prevent us from evaluating the demands multiple robots place on a human teacher.

7.2.1 Synchronous Learning Start Times

First, we present a detailed evaluation of the scalability of the synchronous learning start times approach in which all robots begin learning at the same time. In Sections 7.2.2 and 7.2.3, we compare the performance of the other two techniques – offset learning start times and common policy learning – in the 7-robot beacon homing domain.

Robot Autonomy

Figure 7.2 shows how the level of autonomy, measured as the percentage of autonomous actions versus demonstrations, changes for an individual robot over the course of training. The data presents the average autonomy of robots in the 5-robot beacon homing experiment. The shape of the curve is typical of CBA learning, in which robots begin with no initial knowledge about the task and request many demonstrations early in the training process. These initial demonstrations provide the robot with the experience for handling most commonly encountered domain states. As a result, following the initial burst of demonstration requests, the robot quickly achieves 80–95% autonomous execution. The remainder of the training process then focuses on refining the policy and addressing previously unencountered states. The duration of this learning time is dependent upon the frequency with which novel and unusual states are encountered.

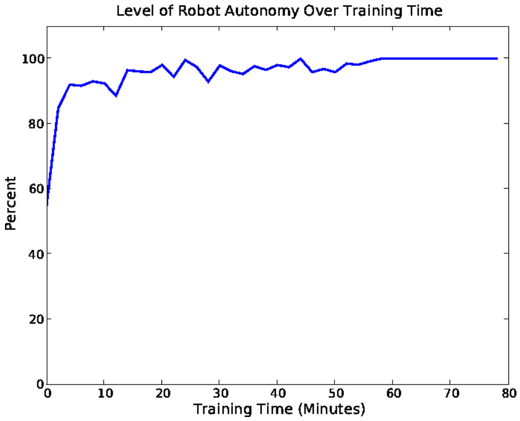


Figure 7.2: Average level of autonomy of a single robot over the course of training.

Number of Demonstrations

Figure 7.3 shows how the number of demonstrations performed by the teacher on average for each robot, and in total for each experiment, changes with respect to the number of robots. As the number of robots grows, we observe a slight increase in the number of demonstrations required per robot. This possibly surprising increase is due to the fact that, although the number of state features in the representation of our domain does not change, the *range* of possible feature values does.

Specifically, in an N-robot experiment, the value of features representing the number of

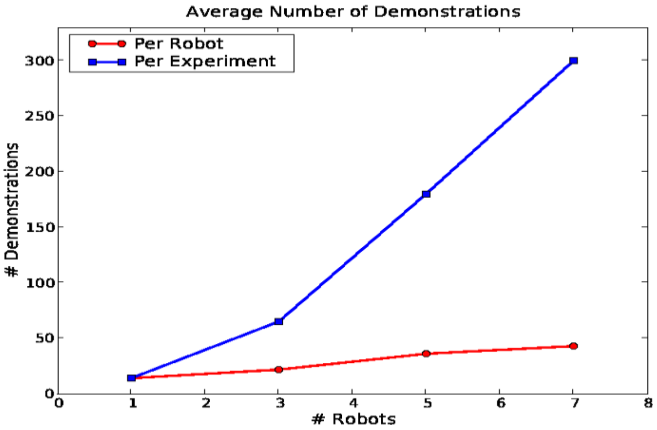


Figure 7.3: Average number of demonstrations performed by the teacher for each robot and in each experiment.

robots located at a beacon, n_i, has the range $[0,N]$. As a result, extra demonstrations are required in the presence of a greater number of robots to provide guidance in the additional states. While similar effects are present in many domain representations, state features can often be designed or modified in such a way that their range is independent of factors such as the number of robots. For example, in the beacon homing domain this could be achieved by converting n_i to a boolean feature that indicates whether the beacon's capacity has been reached or not.

The total number of demonstrations required for each experiment grows at a nearly linear rate with respect to the number of robots. The overall number of demonstrations that the teacher must performed has a significant effect on the overall training time, as discussed in the next section. Seven robots require a total of approximately 300 demonstrations to learn the task.

Training Time

Figure 7.4 presents the change in the overall experiment training time with respect to the number of robots. Importantly for the scalability of *flexMLfD*, the data shows a strongly linear trend, with seven robots requiring just over 1.5 hours to train. This result is significant as it suggests that this approach will continue to scale linearly to even larger tasks. In the following sections, we examine the factors that contribute to the training time, such as the number of demonstrations and demonstration delay.

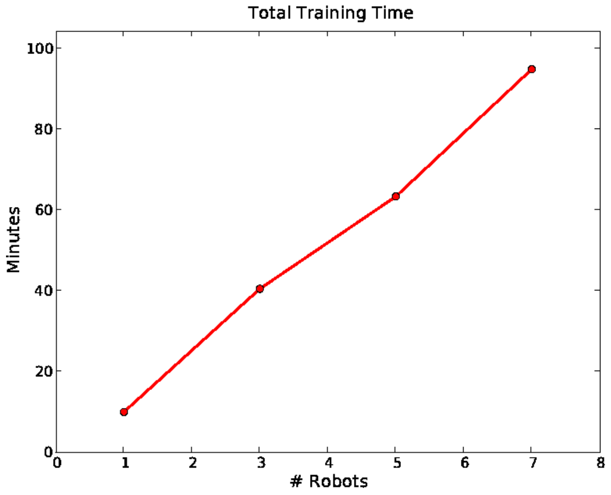


Figure 7.4: Total training time with respect to number of robots.

Attention Demand on the Teacher

In addition to the overall training time and number of demonstrations, it is important to understand the demands that multiple robots place on the teacher. The teacher experiences the greatest number of demonstration requests during the earliest stages of learning, possibly from multiple robots at the same time. To evaluate the demand on the teacher's attention during this most laborious training segment, we calculate the longest continuous period of time during which the teacher has at least one demonstration request pending. This value provides insight into the degree of effort that is required from the teacher.

Figure 7.5 plots the duration of the longest continuous period of demonstration requests for each experiment. The data shows that the length of this time period grows quickly, possibly exponentially, with the number of robots. In experiments with only a single robot, demonstration requests last only a few seconds at a time; as soon as the teacher responds to the request, the robot switches to performing the demonstrated action. As the number of robots increases, however, so does the number of simultaneous requests from multiple robots. In the 7-robot experiment, this results in a 3.5 minute uninterrupted segment of demonstration requests for the teacher. In Section 7.2.2, we examine an alternate approach to teaching multiple robots in which novice learners are added incrementally, reducing the demand for demonstrations early in the training process.

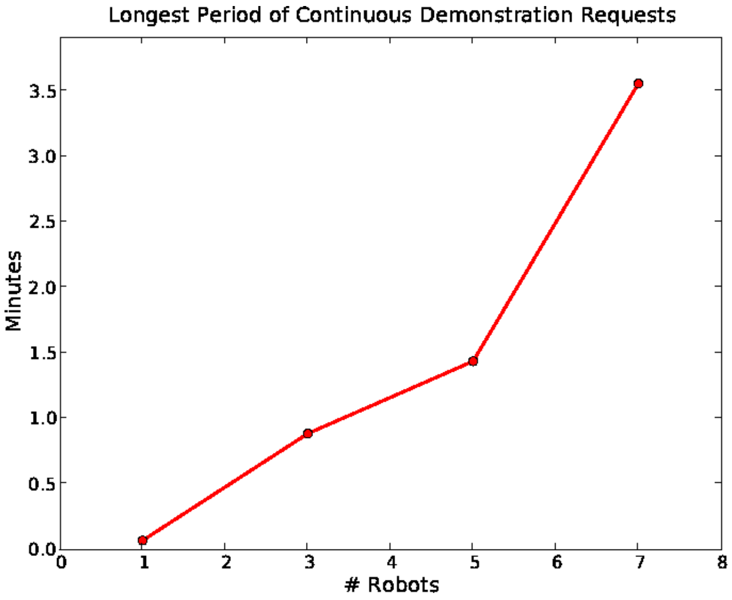


Figure 7.5: Attention demand on the teacher.

Additionally, we examine the total time per experiment that multiple demonstration requests are pending. Figure 7.6 presents a set of bar graphs showing the distribution of the number of simultaneous requests for each experiment. This data indicates that for all experiments, the teacher spends the greatest percentage of time with only a single demonstration request. However, the teacher spends over 3 minutes in the 5-robot experiment, and over 13 minutes in the 7-robot experiment, faced with multiple queries. This growing number of simultaneous queries has a significant impact on demonstration delay, the amount of time that passes between the robot's initial request and the teacher's response.

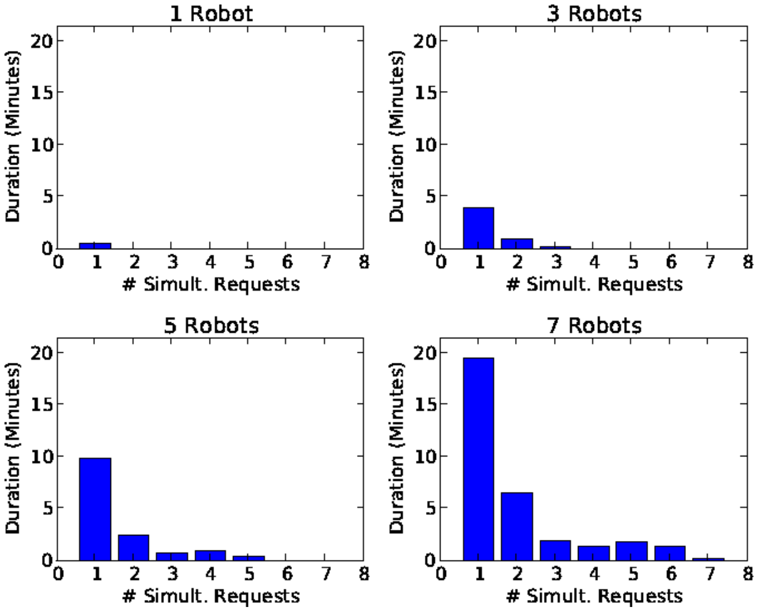
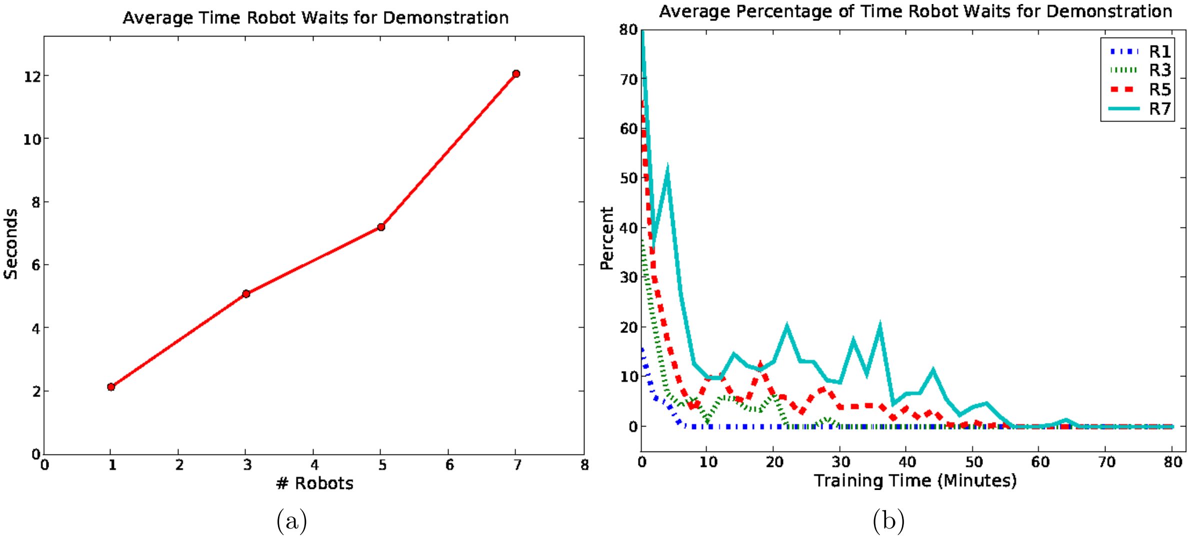


Figure 7.6: Histograms showing the distribution of the number of simultaneous requests for each experiment.

Demonstration Delay

As discussed in the previous section, simultaneous demonstration requests from multiple robots become common as the number of robots increases. As a result, robots often must wait while the teacher responds to other robots. Figure 7.7(a) shows that the average time a robot spends waiting for a demonstration grows with the number of learners from only 2 seconds for a single robot to 12 seconds for seven robots. Figure 7.7(b) plots the percentage of time a robot spends waiting on average for a demonstration over the course of training. Not surprisingly, we observe that the demonstration delay is greatest early in the training process when the teacher is most busy with initial demonstration requests. Staggering the



(a) (b)

Figure 7.7: (a) Average amount of time a robot spends waiting for a demonstration response from the teacher. (b) Average percentage of time a robot spends waiting for a demonstration over the course of training.

times at which novice robots are introduced to the task may help to alleviate this problem by reducing the demand of the initial training phase on the teacher.

Evaluation Summary

In summary, our case study demonstrates the feasibility of a single person training up to seven robots using the *flexMLfD* learning framework. We believe this is the first experiment examining demonstration learning at this scale. Additionally, we find promising trends for even larger experiments. Particularly significant is that the total training time grows linearly with the number of robots. Somewhat unsurprisingly, we also found that increasing the number of robots also significantly increases the workload of the teacher, as measured by the number of pending demonstration requests. In our evaluation, we show that this in turn impacts demonstration delay and robots must spend more time waiting for the teacher's response. While this has no negative impact on learning in the presented domain, delay may impact performance in other tasks.

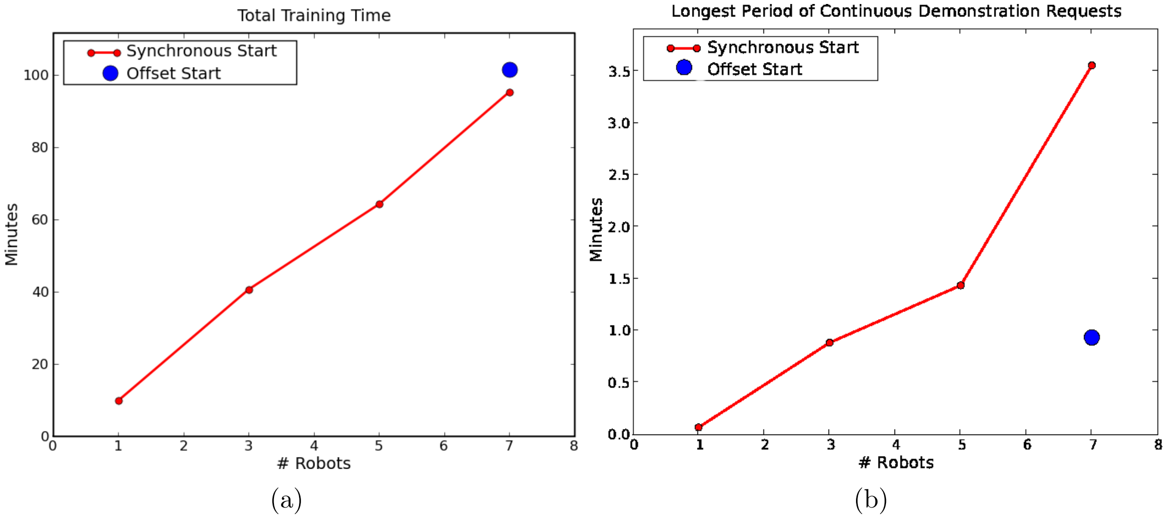
Based on the presented case study, we find that no absolute upper bound exists on the number of robots that can be taught at the same time. The maximum number of robots used in the experiments, seven, represents our own limitation in terms of time and the number of available robots, not a limitation of the algorithm. Furthermore, insights gained in this evaluation provide guidelines for the development of future applications for CBA learning. For example, our knowledge of the trend in overall training time requirements can be used to limit the number of robots in other applications for which the availability of an expert teacher is limited to some fixed time. Similarly, the number of robots in other domains may be affected by the amount of time a robot is allowed to remain idle while waiting for a demonstration.

7.2.2 Offset Learning Start Times

Analysis of the synchronous start times approach presented in the previous section shows that when multiple robots begin learning a new task at the same time, they place great demand on the teacher for demonstrations early in the training process. This, in turn, leads to longer demonstration delays for each robot. In this section, we examine an alternate approach in which novice robots are introduced incrementally. Each new robot is added once all previous learners have gained proficiency at the task and are able to act autonomously approximately 80% of the time. We refer to this learning process as *offset start time* learning.

We compare the performance of offset start time learning to the synchronous start time approach in the 7-robot beacon homing domain. Note that as we add each additional robot learner, the range of domain states experienced by previously active robots expands. For example, a single robot alone can be taught to navigate to various beacons, but without the presence of other robots it will never learn the collaborative aspects of the task. As each additional robot is introduced to the domain, the robot will learn how to act in the presence of others.

Figure 7.8 shows the effect of offset start time learning on total training time and attention demand on the teacher. Figure 7.8(a) shows that by offsetting the learning start time of the robots, the total learning time of the experiment increases by approximately 12%. This increase is due to the fact that robots experience novel states over a longer period of time. Since the task itself remains unchanged, the total number of demonstrations required to teach all seven robots their individual policies remains approximately the same (284 demonstrations using offset start times compared to 298 using the synchronous start times approach). However, since the demonstrations are disbursed over a longer period of time, the attention demand on the teacher is significantly reduced, as shown in Figure 7.8(b). The longest period of continuous demonstration requests falls to approximately 1 minute, compared to 3.5 minutes in the synchronous learning start case, and the average time a robot waits to receive a demonstration falls from 12 seconds to 4.5 seconds.



(a)

(b)

Figure 7.8: Comparison of the synchronous and offset learning start time approaches. (a) Total training time. (b) Attention demand on the teacher.

In summary, by offsetting the learning start times of the robots we are able to distribute demonstration requests over a longer time period. This leads to a slight increase in the overall learning time, but reduces the number of simultaneous demonstration requests the teacher receives, which in turn leads to faster response times, reducing the demonstration delay.

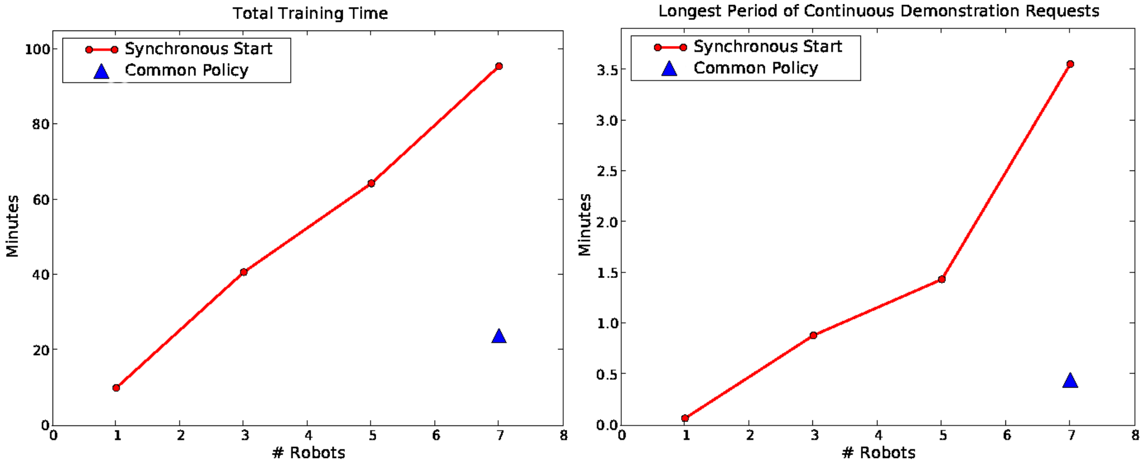
7.2.3 Common Policy Learning

In the standard formulation of the *flexMLfD* learning approach, the teacher provides each robot with an individual set of demonstrations from which a unique policy is derived. This generalized approach is highly suitable for domains in which robots perform different roles and functions. However, in some domains, as in our experiments, the same behavior may be desirable for each robot. In such cases, teaching the same policy to multiple robots results in a large number of redundant demonstrations. To address this case, we propose that all robots learning the same task learn a single, *common*, policy by consolidating all demonstration data. The sharing of information can occur by collecting all data within a single dataset, or by freely exchanging each demonstration among all robots so as to maintain a distributed set of identical policies. In this section, we evaluate the performance of the common policy approach using the 7-robot beacon homing domain.

Note that common policy learning in domains independent robots, results in the same final policy as if each of those robots was taught alone. Using multiple robots in this case may speed up learning since uncommon states are more likely to be encountered with many learners. Common policy learning in domains with non-independent robots, as in our example of the beacon homing domain, differs from training and replicating a single robot policy as it additionally allows the robots to experience the collaborative aspects of the task.

Using this technique, the teacher was required to perform a total of only 44 demonstrations, compared to nearly 300 total demonstrations previously required for this task. Figure 7.9(a) shows that the overall training time of the experiment is similarly reduced to only 23 minutes, compared to the 95 minutes for the standard approach. In fact, while seven robots learning a common policy require more of the teacher's time than a single learner, they require less time than three robots learning individual policies.

Figure 7.9(b) shows that a common policy similarly reduces the attention demand that seven robots require of the teacher. This effect can be attributed to the fact that frequently a demonstration performed for one robot addresses the queries of other currently waiting robots. An additional effect of this occurrence is that the average waiting time of the common policy approach is reduced from 12 seconds to 1.2 seconds.



(a) (b)

Figure 7.9: Comparison of the single common policy and multiple individual policy *flexMLfD* learning approaches. (a) Total training time. (b) Attention demand on the teacher.

7.3 Chapter Summary

This chapter presented a case study analysis of the scalability of the *flexMLfD* learning framework in a beacon homing domain using up to seven AIBO robots. We compared three approaches to teaching multiple robots at the same time: synchronous learning start times, offset learning start times, and common policy learning. Learning performance was evaluated with regard to the number of demonstrations required to learn the task, the demands for time and attention placed on the teacher, and the delay that each robot experiences in obtaining a demonstration.

Our results show that in the synchronous learning start approach, demands on the teacher and robots grow at a roughly linear rate with respect to the number of robots. By offsetting the learning start times of the robots, demonstration requests can be distributed over a longer time period, reducing the attention demand on the teacher and demonstration delay. In the shared learning scenario, the training time and number of demonstrations can be significantly reduced by learning a single common policy. Most importantly, our analysis indicates that no strict upper bound exists on the number of robots due to the algorithm's limitations. Instead, the number of robots is likely to be limited by real-world factors particular to each learning domain, such as the amount of time the teacher is able to invest in training.

Chapter 8

Related Work

Demonstration-based learning techniques are described by a variety of terms within the published literature, including learning by demonstration (LbD), learning from demonstration (LfD), programming by demonstration (PbD), learning by showing, learning by watching, learning from observation, behavioural cloning, imitation and mimicry. While the definition for some of these terms, such as imitation, have been loosely borrowed from other sciences, the overall use of these terms is often inconsistent or contradictory across articles.

Within this thesis, we refer to the general category of algorithms in which a policy is derived based on demonstrated data as *learning from demonstration*. In a recent survey [5], we identify the defining features of these techniques and present a detailed characterization of existing learning from demonstration approaches. In this chapter, we present a subset of the survey's organizational structure and discuss where our work fits with relation to existing approaches.

8.1 Characterization of Demonstration Learning Approaches

There are two aspects of learning from demonstration which are common among all applications to date. One is the fact that a teacher demonstrates execution of a desired behavior. The other is that the learner is provided with a set of these demonstrations, and from them derives a policy able to reproduce the task. What distinguishes approaches is the manner in which each of these problems is structured and solved. In this section, we segment the demonstration learning problem into these two fundamental phases, data gathering and policy learning, and survey different solutions to each.

8.1.1 Data Gathering

In this section, we discuss various techniques for obtaining demonstrations. A demonstration dataset is composed of state-action pairs recorded during teacher executions of the desired behavior. Exactly how demonstrations are recorded, and who performs the demonstrations, varies greatly across approaches.

Learning from Observation

In the *learning from observation* data gathering approach, demonstration examples are obtained by the robot by passively observing the actions of the teacher. As a training method, learning from observation is based on the assumption that the task being learned is best demonstrated by the teacher through independent and uninhibited execution using her own body. This approach is commonly used to demonstrate low-level control functions for high degree of freedom robots, such as humanoids, for which teleoperation of the robot would be difficult. Common applications include object manipulation [51, 67, 89] and interactive games [8, 43].

In this data gathering approach, the burden of interpreting and extracting useful information from the observed demonstration is placed on the robot. This gives rise to the *correspondence problem* [24], the challenge of identifying the state and action of the demonstrator and mapping them to the robot's own abilities with the corresponding effect. Accurate and efficient methods for solving this problem have been the focus of extensive research [3, 24, 40, 45, 46, 89].

Most observation-based methods rely on advanced computer vision systems to record the demonstration and recognize the actions performed by the demonstrator [9, 45, 89]. Extracting meaningful information from image sequences is a very challenging problem, and marker-based vision systems have been developed that provide additional information about the demonstrator's movements. Additional data sources have included tactile [27, 89], position [16, 40], and magnetic sensors [27], as well as speech recognition [27, 52, 73].

Learning from Experience

In the *learning from experience* data gathering approach, demonstrations are performed by letting the robot experience the desired actions using its own body, either through manipulation of robot parts or by instructing the robot which of its actions to perform. The robot actively participates in the demonstration, executing the actions selected by the teacher while sensing the environment with its own sensors. This approach allows the robot to directly associate gathered sensory information with one of its own actions, eliminating the correspondence problem faced by observation-based methods.

To perform the demonstration, the expert must have a way to direct the robot's actions. One of the most commonly used methods is teleoperation, which allows the expert to control the robot's actions directly via a joystick, or similar interface, while monitoring the robot's progress [12,13,42,48,74,85]. Most teleoperation approaches to date have made use of human teachers [12,15,61], although several techniques have also examined the use of hand-written control policies [4,35,76]. Teleoperation requires that operating the robot be manageable, and as a result not all systems are suitable for this technique.

Another method for performing the demonstration is teacher following, in which the robot follows a human or robot teacher through the execution of the task [37,60,62]. Targeted at navigation tasks, this approach combines elements of observed and experienced demonstration as the robot must observe and imitate the behavior of the teacher. However, since the robot observes the environment with its own sensors and executes its own actions, finding the correspondence between the teacher's behavior and the robot's is not required.

In this thesis, we utilize a learning from experience approach for performing demonstrations. During demonstration, the robot's behavior is controlled through a graphical user interface that enables the teacher to select from among the robot's available actions.

Demonstration Selection

The approaches described above characterize demonstrations by how they are performed. It is also important to consider how demonstrations are *selected*. Most demonstration learning approaches rely on the teacher to select all demonstrations [9,74,76]. By placing the robot in a particular environment, providing execution examples, and turning demonstration learning on and off the teacher controls the flow of information.

An alternate approach to demonstration selection is *interactive demonstration*, in which data is acquired through an active dialog between robot and teacher. In this approach the robot is no longer a passive receiver of demonstrated examples, but a collaborator that provides feedback to the teacher about the learning process, indicating uncertainty or even asking questions about the task. This approach forms a natural extension of demonstration learning as it builds upon an existing interaction framework, while allowing the robot to help guide the learning process. Note that interactive demonstration is not mutually exclusive from the previous two data gathering techniques. Interactive demonstration describes the method by which demonstrations are selected, and not the way they are performed.

A number of algorithms have been developed that enable a robot to learn through interaction with other autonomous robots and human teachers. In the context of reinforcement learning, Clouse presents the 'Ask For Help' framework [25]. This approach enables a robot to request advice from other robots when it is "confused" about what action to take, an event characterized by relatively equal quality estimates for all possible actions in a given state.

Nicolescu and Mataric [62] present a learning framework based on demonstration, generalization and teacher feedback, in which training is performed by having the robot follow a human and observe its actions. A high-level task representation is then constructed by analyzing the experience with respect to the robot's underlying capabilities. The authors also describe a generalization of the framework that allows the robot to interactively request help from a human in order to resolve problems and unexpected situations. This interaction is implicit as the robot has no direct method of communication; instead, it attempts to convey its intentions by communicating though its actions.

Lockerd and Breazeal [11,52] demonstrate a robotic system where high-level tasks are taught through social interaction. In this framework, the teacher interacts with the robot through speech and visual inputs, and the learning robot expresses its internal state through emotive cues such as facial and body expressions to help guide the teaching process. The outcome of the learning is a goal-oriented hierarchical task model. In later work [82], the authors examine ways in which people give feedback when engaged in an interactive teaching task. Although the study's focus is to examine the use of a human-controlled reward signal in reinforcement learning, the authors also find that users express a desire to guide or control the robot while teaching. This result supports our belief that, for many robotic domains, teleoperation provides an easy and intuitive human-robot communication method.

Grollman and Jenkins present the Dogged Learning algorithm [35], a confidence-based learning approach for teaching low-level robotic skills. In this algorithm, the robot indicates to the teacher its certainty in performing various elements of the task. The teacher may then choose to provide additional demonstrations based on this feedback. While similarly motivated, our work differs from the Dogged Learning algorithm in a number of ways, most important of which are our use of classification instead of regression in policy learning, and our algorithm's ability to adjust the confidence threshold to the data instead of using a fixed value. Inamura et al. [42] present a similarly motivated method based on Bayesian Networks that is limited to a discretely-valued feature set.

In addition to learning from demonstration, several other areas of research have also explored algorithms for demonstration selection. Within machine learning research, active learning [10,26] enables a learner to query an expert and obtain labels for unlabeled training examples. Aimed at domains in which a large quantity of data is available but labeling is expensive, active learning directs the expert to label the more informative examples with the goal of minimizing the number of queries.

In this thesis, we contribute an interactive, mixed-initiative approach to learning from demonstration that enables both the robot and the teacher to select demonstration examples.

8.1.2 Policy Learning

Once the demonstration data has been obtained, the robot must learn a policy that enables it to reproduce the desired behavior. Learning from demonstration research has seen the development of three core approaches to deriving policies from demonstration data. Learning a policy can involve learning an approximation of the state to action mapping (*mapping function*), or learning the model of the world dynamics and deriving a policy from this information (*system model*). Alternately, a sequence of actions can be produced by a planner after learning a model of action pre- and post-conditions (*plans*). Across all of these learning techniques, minimal parameter tuning and fast learning times requiring few training examples are desirable.

Mapping Function

The mapping function approach to policy learning calculates a function that approximates the state to action mapping for the demonstrated behavior. The goal of this type of algorithm is to reproduce the underlying teacher policy, which is unknown, and to generalize over the set of available training examples such that valid solutions are also acquired for similar states that may not have been encountered during demonstration.

The details of function approximation are influenced by many factors. These include whether the state input and action output are continuous or discrete; whether the generalization technique uses the data directly at execution time or uses the data to approximate a function prior to execution time; whether it is feasible or desirable to keep the entire demonstration dataset around throughout learning; and whether the algorithm updates online.

In general, mapping approximation techniques fall into two categories depending on whether the prediction output of the algorithm is discrete or continuous. Classification techniques are used to produce discrete output. Example classification methods applied to learning from demonstration have included k-Nearest Neighbors [74], Bayesian Networks [11] and Hidden Markov Models [28]. Regression techniques produce continuous output. Regression algorithms applied to demonstration learning for robotic systems include Locally Weighted Regression [12, 59, 78], Gaussian Mixture Regression [13], and On-Line Gaussian Processes [36]. Classification and regression algorithms typically learn aggressively and are able to generalize from a small number of examples. A common assumption held by these approaches is that for any world state their exists a single best action.

In this thesis, robot policies are represented and learned using classification. Additionally, we contribute a classifier-independent algorithm that enables states to be mapped to a choice of multiple actions.

System Model

The system model approach to policy learning determines a state transition model for the world from which it then derives a policy. This approach is typically formulated within the structure of reinforcement learning (RL) [80]. The standard reinforcement learning system assumes that the world can be described by a finite set of states, and that the robot is limited to taking one of a finite set of actions at each discrete timestep. At each learning timestep, the robot observes its state in the world and chooses an action to execute. After completing the action, the robot receives a reinforcement signal, or reward, that reflects the goodness of the action or the resulting state. The robot's goal is to learn a mapping from states to actions that maximizes the robot's reward over time. Among the many extensions to the standard reinforcement learning algorithm, a wide range of approaches have been developed for applying RL to continuous state and action domains [56, 58, 77, 80, 84].

Within the context of reinforcement learning, demonstration can be seen as a source of reliable information, or advice, that can be used to accelerate the learning process [54,63,71]. A number of approaches for taking advantage of this information have been developed, such as deriving or modifying the reward function based on the demonstration [1,7,25,65,82], and using the demonstration experiences to prime the robot's value function or model [68,75,81]. [77] propose an alternate approach for accelerating learning in domains with sparse rewards.

Instead of attempting to model expert behavior by rewarding similar actions, they allow the expert to demonstrate the execution of the task and expose the robot to areas of the state space where the reward is non-zero. Bootstrapped with this information, the robot is then able to quickly learn the task under autonomous execution.

Combining demonstration with an exploration-based learning method allows the robot to improve upon the demonstrated behavior and surpass the expert in performance. The benefit of this approach is that the robot is not bound by the abilities of the teacher, and the teacher is not required to make a perfect demonstration. Exploration, however, can be expensive in terms of time or hardware costs for many robotic tasks.

Plans

Within the planning framework, a policy is represented as a sequence of actions that lead from the initial state to the final goal state. Actions are often defined in terms of pre-conditions, the state that must be established before the action can be performed, and post-conditions, the state resulting from the action's execution. Unlike other learning from demonstration approaches, planning techniques frequently rely not only on state-action demonstrations, but also on additional information in the form of annotations or intentions from the teacher [31, 32, 85]. Demonstration-based algorithms differ in how the rules associating pre- and post-conditions with actions are learned, and whether additional information is provided by the teacher.

Nicolescu and Mataric [62] introduce a plan-based framework in which training is performed by having the robot follow a human and observe its actions. A high level task representation is constructed by the algorithm by analyzing this experience with respect to the robot's abilities. Veeraraghavan and Veloso [86] present an algorithm for learning generalized plans that represent sequential tasks with repetitions. In this framework, a humanoid robot is taught the repetitive task of collecting colored balls into a box based on two demonstrations. Rybski et al. [73] present a system in which demonstration of the desired task is performed through speech dialog. The teacher presents the robot with a series of conditional statements which are processed into a plan. The robot is additionally able to verify any unspecified parts of the plan through dialog.

8.2 Multi-Robot Learning

Learning from demonstration approaches have almost exclusively focused on the problem of a single robot being taught by a single teacher. A handful of projects, however, have examined the application of demonstration learning in the context of multi-agent software systems. Oliveira and Nunes [63] and Clouse [25] present algorithms that allow agents to request advice from other similar agents within a reinforcement learning framework. Upon request, each agent provides advice to teammates based on its experiences and learned model. Price and Boutilier [68] present a multiagent system in which novice agents learned from passively observing expert agents in the environment. Each learning agent is limited to observing the actions of others, and no explicit teaching occurs. By observing a mentor, the reinforcement learning agent is able to extract information about its own capabilities in, and the relative value of, unvisited parts of the state space. However, the task of an observed agent may be so different that the observations provide little useful information for the learner, in which case direct imitation of this expert must be avoided by the algorithm.

Alissandrakis et al. [2,3] present a general framework that enables a robotic agent to imitate another, possibly differently embodied, agent through observation. Using this framework, the authors demonstrate the transmission of skills between individuals in a heterogeneous community of software agents. Their results indicate that transmission of a behavior pattern through a chain of agents can be achieved despite differences in the embodiment of some agents in the chain. Additional results indicate that groups of mutually imitating agents converge to a common and shared behavior.

The above methods study imitation from the perspective of a community of agents, where a single agent seeks advice from other members of its group. A different approach is taken in the study of coaching [70], in which an external coach agent provides advice to a team of agents in order to improve their performance at a task. The coach has an external, often broader, view of the world and is able to provide advice to the agents, but not control them. The agents must decide how to incorporate the coach's advice into their execution; they can not request help from the coach.

Work presented in this thesis differs from existing multi-robot techniques in that it enables a single human to simultaneously train multiple robots. Additionally, we place no restrictions on the robots in terms of embodiment or behavior, allowing each robot to learn its individual policy or task.

8.3 Multi-Robot Control

Interfaces for the interaction and control of multiple robots have been studied in the human-robot interaction (HRI) community for many robotics applications.

Humphrey et al. [41] present a relational display designed for the control of a variable number of robots, which was successfully used to teleoperate up to nine simulated robots in a bomb defusing task. Fong et al. [30] present HRI/OS, a software framework for establishing interaction and dialog in human-robot teams. Targeted at execution of operational tasks, such as resource collection, this system supports a variety of user interfaces and robot platforms through an extensible API, making it applicable to a wide range of applications.

Wang and Lewis [90] present a study evaluating how the degree of operator control affects task performance in multi-robot tasks. The authors conclude that mixed initiative teams of robots performed more successfully at a simulated urban search and rescue task than either fully autonomous or manually controlled teams. Similar results are supported by other studies in adjustable autonomy research, motivating the use of partial autonomy in our multi-robot demonstration learning framework.

Other research has examined the human side of the interaction, examining the effects of various design elements on user performance [34, 38]. For example, closely related to our problem of multi-robot demonstration learning is an interface proposed by Glas et al. [33] for the control of multiple social robots.

In learning from demonstration, the role of the teacher is generally similar to that of the operator in the systems described above. The teacher oversees the robots, provides robot commands in the form of actions, and identifies errors that the robot can not detect on its own. However, the above systems provide interfaces only for robot control and navigation, and do not include a learning component. In particular, we are not aware of any system designed to address multi-robot interaction and control in the context of learning from demonstration. *The interaction and control interface presented in this thesis combines policy learning, demonstration requests, adjustable autonomy, and correction capabilities within a single software system.*

Chapter 9

Conclusion and Future Work

This chapter reviews the dissertation's scientific contributions and presents several promising future research directions that build on this thesis.

9.1 Contributions

This thesis makes the following scientific contributions:

- **Confidence-Based Autonomy single-robot demonstration learning algorithm**

  The Confidence-Based Autonomy algorithm enables a human user to train a task policy through demonstration. The algorithm consists of two components. The Confident Execution component enables the robot to select demonstrations in real time as it interacts with the environment, using confidence and distance thresholds to target states that are unfamiliar or in which the current policy action is uncertain. To enable the robot to request a demonstration at any time, we assume that the robot can pause between executions of consecutive actions. The Corrective Demonstration component allows the teacher to additionally perform corrective demonstrations when an incorrect action is selected by the robot. The teacher retroactively provides demonstrations for specific error cases instead of attempting to anticipate errors ahead of time. Combined, these techniques provide a fast and intuitive approach for policy learning, incorporating shared decision making between the learner and the teacher.

- **Algorithm for identifying, representing and enacting equivalent action choices**

  Regions of inconsistent demonstrations are frequently encountered in learning from demonstration due to the existence of multiple applicable actions or sensor noise. This thesis introduces a classifier-independent algorithm based on Confidence-Based Autonomy that enables choices between multiple actions to be represented explicitly within the robot's policy through the creation of option classes.

- **A task-independent and robot-independent interface for demonstration learning**

  This thesis contributes a detailed description of the fully implemented, robot-independent and task-independent demonstration learning software system used for the evaluation of all presented algorithms. A detailed description of the software system in Appendix A should enable future researchers to build upon this work and develop similar systems.

- ***flexMLfD*, a robot-independent and task-independent multi-robot demonstration learning framework**

  This thesis contributes *flexMLfD*, the first multi-robot demonstration learning framework. Our approach is based on the Confidence-Based Autonomy algorithm, which establishes a general state and action representation and provides a means for single-robot policy learning through adjustable autonomy. Using an independent instance of the CBA algorithm, each robot acquires its own set of demonstrations and learns an individual task policy. The unique feature of the CBA algorithm that enables it to be applied to multi-robot learning is the Confident Execution component, which enables each robot to regulate its own autonomy, and to pause execution when faced with uncertain situations.

- **Three approaches to teaching collaborative multi-robot tasks**

  Solutions to many complex tasks require the collaboration of multiple robots. Based on the *flexMLfD* multi-robot demonstration learning framework, this thesis examines methods for teaching emergent multi-robot coordination, in which the solution to the shared multi-robot task emerges from the complementary actions performed by robots based on their independent policies. We contribute three approaches to teaching collaborative behavior based on different information sharing strategies: implicit coordination, coordination through active communication, and coordination through shared state.

- **Case study analysis of the scalability of *flexMLfD***

  Controlling and teaching multiple robots at the same time is a challenging task. This thesis contributes a case study analysis of the scalability of the *flexMLfD* learning framework in a beacon homing domain using up to seven AIBO robots. We compare three approaches to teaching multiple robots at the same time: synchronous learning start times, offset learning start times, and common policy learning. Learning performance was evaluated with regard to the number of demonstrations required to learn the task, the demands for time and attention placed on the teacher, and the delay that each robot experiences in obtaining a demonstration.

- **Extensive evaluation of all learning methods**

  This thesis presents detailed evaluations of all presented algorithms in multiple simulated and real-world domains. Evaluations span multiple robotics platforms (Sony QRIO and AIBO), incorporate different number of robots (from 1 to 7), and utilize a wide variety of state and action representations.

9.2 Future Directions

Over the course of my PhD, I have been involved with a number of robotics research projects outside of learning from demonstration, including robot soccer (Appendix B), interactive human-robot teams [53], and evolving long-term human-robot interaction [17]. These projects utilized traditional robot development approaches, in which groups of researchers coded all robot capabilities by hand without the use of learning from demonstration. While I believe that certain components of these projects required manual coding (e.g. development of novel learning algorithms or software infrastructures), much work was also devoted to applications that are highly suitable for demonstration learning. This included the development of a wide range of robot actions and behaviors, such as navigation, object manipulation, person following, obstacle avoidance and multi-robot coordination. In this section, we present some of the lessons learned from the development and evaluation of the confidence-based demonstration learning methods introduced in this thesis, and discuss key directions for future research that can enable learning from demonstration to be applied to the broad range of applications mentioned above.

The policy learning algorithms presented in this thesis target a specific class of domains – those consisting of high level tasks with a discrete action space and the ability for the robot to pause to request a demonstration. Within this broad class of domains, we have shown our algorithms to be robust and able to learn a wide range of different tasks. Our long-term goal is to extend confidence-based learning to an even wider range of applications by building upon the presented algorithms, relaxing existing assumptions, and incorporating new features. One possible future application is robot soccer, a complex multi-robot domain that combines low level continuous control and navigation, high level decision making, and real-time multi-robot coordination. Further research is several key directions is required to enable a task of this complexity to be learned from demonstration.

- **Enhanced demonstration interface** As part of this thesis, we contribute a graphical user interface for information exchange and interaction between the robot and teacher. The interface displays the robot's state, composed of locally observed and communicated information, and enables the teacher to select actions during demonstration. One of the lessons learned during this research, is that communicated information shared wirelessly between robots can be difficult for the teacher to track. Since communication is imperceptible and instantaneous, the teacher must rely heavily on the GUI to monitor the values of communicated data. If a large amount of data is communicated between robots, and if data values change frequently, maintaining an accurate mental model of the robot's state becomes difficult for the teacher. An important direction for future work is the further development and evaluation of interfaces for interactive demonstration learning, including user studies and exploration of methods for visualizing communicated data.
- **Pause-less demonstration** In many applications it is not possible or practical to perform demonstrations while the robot is stopped or paused. This may be because the robot is unable to stop safely, or because stopping changes the robot's state (e.g. velocity). In highly dynamic environments, pausing may simply be impractical if the environment changes quickly, leaving too little time for the teacher to perform a demonstration for the relevant state. For example, robot soccer games are typically so fast-paced that the game state can change significantly after just a three second delay. To enable learning is these kinds of domains, a demonstration method is required that does not rely on pausing. One possible solution would be to teach the robots entirely through correction; in the resulting approach, demonstration data would be provided in real time during execution, but the robots are likely to make many mistakes while learning. Another possible approach is to enable demonstrations and corrections to be performed retroactively based on records of past actions, similar to the critiquing technique presented by Argall et al. in [4].
- **Learning of continuous actions** Algorithms presented in this thesis are designed for learning high level task policies with discrete action spaces. In many domains, the discrete actions encode low level behaviors such as navigation (e.g. turn left) or manipulation (e.g. pass ramp). Within this thesis, all low level actions were hand-coded by specifying the exact sequence of movements required to achieve the desired effect (e.g. number of degrees to turn, or arm joint angles necessary to pick up and drop a ball). A number of algorithms have been proposed to enable these kinds of low-level actions to also be learned from demonstration [4,35]. An interesting direction for future

work is to eliminate the need for hand-coded actions entirely within the demonstration learning system, either by combining our work with other proposed approaches, or by extending confidence-based interactive learning to continuous domains.

- **High-dimensional state and action spaces** Algorithms presented in this thesis, as most policy learning algorithms, are provided with exactly the data needed to learn the task, nothing more and nothing less. Complex robotic systems, however, may have hundreds of possible state features (ranging from raw sensor readings to many calculated features such as person recognition data) and dozens of actions. The Confidence-Based Autonomy algorithm relies on classification for underlying policy learning, enabling it therefore to learn to generalize over irrelevant features. However, learning this generalization may require a large number of demonstrations. Methods for reducing the size of the state and action space, either through configuration files, interaction with the user, or through automated means such as feature selection and dimensionality reduction, are required to enable demonstration learning algorithms to be applied in a general setting and to more complex domains.

- **Multi-robot demonstration sharing** In multi-robot domains, many behavioral elements of the task may be common across multiple robots. For example, in the QRIO ball sorting domain, each robot performed a different sorting behavior (differentiated by ball color), but the behavior to be performed when the ball queue was empty was common to both robots. We frequently encounter common task elements such as this in multi-robot domains. In such cases, learning efficiency can often be significantly improved by enabling robots to share this type of common information among themselves by teaching each other. Automated approaches for identifying common information to be shared remains a challenging research problem.

- **Improvement beyond teacher performance** The policy learning algorithms presented in this thesis are based on the assumption that the teacher is an expert at the task being taught. Policy learning relies entirely on training data obtained from the teacher and aims to reproduce the teacher's behavior as closely as possible. As a result, performance is inherently limited by the quality of the information provided in the demonstration dataset, and any errors made by the teacher are replicated by the robot. An important direction for future work is to extend learning to enable the robot's performance to surpass that of the teacher. Possible approaches include incorporating a high-level feedback [4] or a reward signal [82] from the teacher, as well as filtering noisy or inaccurate demonstrations.

Appendix A

Confidence-Based Autonomy Software System

The Confidence-Based Autonomy algorithm establishes a general state and action representation and provides a means for single-robot policy learning through adjustable autonomy. In this section, we describe the software system and interface designed around CBA to enable interaction between the robot and teacher. Figure A.1 presents an overview of the demonstration learning software system, which consists of three software modules: the Learning Control Interface (LCI), the policy learner, and the robot's onboard software controller.

The LCI is a task-independent and robot-independent software component that provides a central control point for learning. It contains the CBA algorithm and manages the interaction between all system components. Additionally, the LCI provides a graphical user interface (GUI) for interaction between the learning software and the teacher.

The policy learning component can be viewed as a black box containing the classifier of the teacher's choosing, for which the LCI provides a generic interface. Communication between the LCI and policy learner consists of three types of information. The LCI provides demonstration training data to the learner, as well as classification queries using the robot's current state. For each query, the policy returns its highest confidence action, the action selection confidence and decision boundary information that is used by the CBA algorithm.

Our system makes no assumptions about the physical embodiment and onboard software architecture of the robot beyond the following requirements: 1) the robot's onboard controller is able to establish a TCP connection with the LCI for data transfer; 2) the robot's onboard controller is able to perform a set of predetermined actions the execution of which can

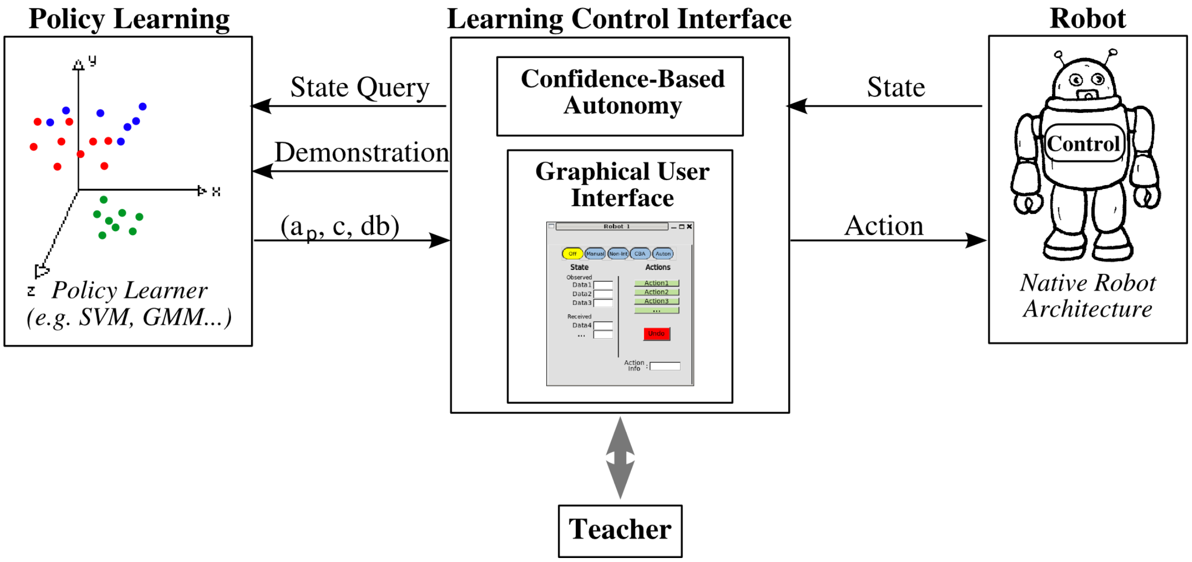

Policy Learning
y
x
z
Policy Learner
(e.g. SVM, GMM...)
State Query
Demonstration
(a_p, c, db)
Learning Control Interface
Confidence-Based Autonomy
Graphical User Interface
Robot 1
Off
Manual
Non-Int
CBA
Auton
State
Actions
Observed
Data1
Data2
Data3
Received
Data4
...
Action1
Action2
Action3
...
Undo
Action Info :
State
Action
Robot
Control
Native Robot Architecture
Teacher

Figure A.1: Confidence-Based Autonomy demonstration learning software system.

be triggered by the LCI. Any robotic, or even software, system that meets these general requirements can be used for demonstration learning. Communication between the LCI and the robot consists of the exchange of state and action information.

All communication between software components occurs over Transmission Control Protocol (TCP) sockets. The modular structure of the presented learning system allows individual components to be switched in and out freely, which not only enables the user to apply the base system to a variety of robotic platforms and tasks, but also allows for independent development and testing of each system element.

For each task, we assume the existence of basic robot abilities that are required for the task, specifically sensor data required for observation of the robot's state, and basic action primitives that can be combined to perform the overall task. To enable interaction between the robot and teacher, a graphical user interface is used to display the robot's state information, and to provide the teacher with the choice of actions for demonstration. The teacher performs demonstrations by selecting among the available actions in the GUI.

At the beginning of the learning process for each new task, the teacher configures the LCI through an XML configuration file. Information provided in each configuration includes robot information (name, IP, and port), choice of policy learning algorithm (e.g. SVM), and the name of the log file in which a record of demonstrations is maintained. Additionally, the teacher may select from among the robot's abilities the set of state features and actions

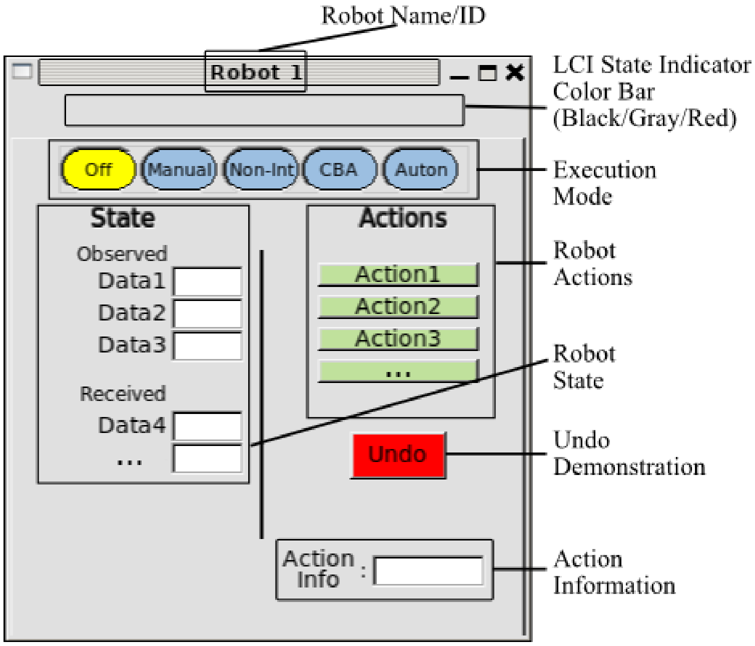

Robot Name/ID
Robot 1
LCI State Indicator
Color Bar
(Black/Gray/Red)
Off
Manual
Non-Int
CBA
Auton
Execution
Mode
State
Observed
Data1
Data2
Data3
Received
Data4
...
Actions
Action1
Action2
Action3
...
Robot
Actions
Robot
State
Undo
Undo
Demonstration
Action
Info
Action
Information

Figure A.2: Screenshot of the LCI graphical user interface.

that are relevant to the task being learned. This selection process speeds up learning by reducing the state representation only to relevant features, and simplifies the user interface for demonstration.

A.1 Graphical User Interface

The graphical user interface of the LCI serves as a two-way communication device between the robot and the teacher. A screenshot of the GUI is shown in Figure A.2 with labels highlighting different regions of the display. The interface displays system information, such as the identification number or name of the associated robot, and the robot's current state. State features are organized by their source, separated into information observed locally through the robot's sensors and information received through communication from remote sources, such as teammates or external sensors. Communicated data can be especially difficult for the teacher to monitor without the GUI interface. The robot state display therefore provides vital information necessary to ensure that the teacher's view of the world matches that of the robot.

Using the GUI, the teacher is able to perform demonstrations by selecting among a set of possible actions for the robot to execute. The GUI also allows the teacher to undo incorrect demonstrations in the case that the wrong action was accidentally selected. The undo operation erases the last demonstration performed from the policy database. However, it does not undo the effects of the incorrect action that was executed by the robot as a result of the mistaken demonstration. An action information display, located in the lower right corner of the GUI, shows the current action being performed by the robot. When the robot is idle and a demonstration request is pending, the display shows the highest confidence policy action as a recommendation to the teacher.

A color bar at the top of the GUI screen changes color depending upon the current status of the robot: not connected (black), connected and executing an action (gray), or waiting for a demonstration (red). This enables the teacher to identify at a glance when the robot requests a demonstration, or if connectivity to the robot is broken. Finally, a set of buttons at the top of the display control the execution mode of the robot, as described in the following section.

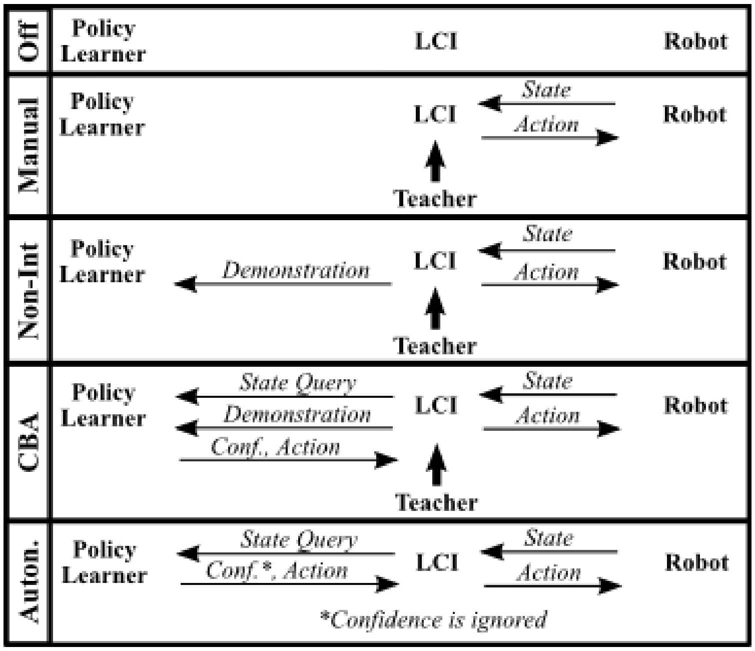
A.2 LCI Execution Modes

The LCI operates in one of five execution modes, which provide varying degrees of control and interaction with the robot. We present a description of each execution mode below. Figure A.3 presents a summary of the interaction between system components for each execution mode.

Off – In this mode the LCI is inactive and no communication occurs between components. This is the starting mode of the system.

Manual – This mode provides the teacher with manual control of the robot's actions, similar to a joystick. The LCI displays the robot's current state and allows the teacher to select actions for execution. The robot executes the specified actions, but no learning takes place. This mode is useful for basic interaction with the robot, such as for testing action performance or for teleoperation.

Non-Interactive – This mode enables the teacher to perform demonstrations without receiving feedback from the LCI regarding when and what demonstrations should be performed. Instead, the LCI requests a demonstration at every learning timestep, regardless of action selection confidence. For each demonstrated action, the LCI communicates the action to the robot for execution, and additionally communicates the state and action pair to the policy learning component as a new training point. This mode enables the teacher to perform long batch demonstration sequences. This can be useful for bootstrapping the learning process,



Off
Policy Learner
LCI
Robot
Manual
Policy Learner
LCI
State
Action
Robot
Teacher
Non-Int
Policy Learner
Demonstration
LCI
State
Action
Robot
Teacher
CBA
Policy Learner
State Query
Demonstration
Conf., Action
LCI
State
Action
Robot
Teacher
Auton.
Policy Learner
State Query
Conf.*, Action
LCI
State
Action
Robot
*Confidence is ignored

Figure A.3: Interaction between the LCI and system components in each execution mode.

such as by demonstrating the complete task once from beginning to end at the start of the learning process.

CBA – In this mode, the LCI uses the CBA algorithm to control robot autonomy and select demonstration. Additionally, it enables the teacher to initiate corrective demonstrations by selecting the correction action in the GUI when an incorrect action executed by the robot is observed. When a correction occurs, the LCI records the new demonstration, communicating it to the policy learner. Note that once initiated, the incorrect action is allowed to complete without interruption; interrupting an action may cause the robot to enter an unstable or unsafe state.

Autonomous – In this mode, robot's actions are controlled by the current learned policy. The robot is fully autonomous, not relying on the teacher for any demonstrations. Each time the robot completes an action, it updates the LCI with its new state information. In turn, the LCI queries the policy to obtain an action and action selection confidence. In this mode, the action selection confidence is ignored, and the policy-selected action is always executed.

Appendix B

RoboCup Standard Platform League Robot Soccer

RoboCup is an international initiative that aims to promote AI and robotics research through organized competitions and conference [6, 50]. Each year, thousands of researchers come together to present the latest research and compete in a number of competitions. The majority of the competitions are organized around the game of robot soccer, in which teams of autonomous robots play against each other. The long-term goal of the RoboCup robot soccer effort is: "*By the year 2050, develop a team of fully autonomous humanoid robots that can play and win against the human world champion soccer team.*"

The robot soccer competitions are organized into multiple leagues, each of which aims to address different technical challenges of the long-term goal. The rules and regulations of each league are designed to drive research in a particular direction, and rules are revised each year in response to research progress.

The Standard Platform League[1] focuses on software and algorithm development. All league participants are restricted to using the same robot platform, and the competition focuses on the development of the software system for these robots, including components ranging from motion generation to sensor processing and behavior execution.

Prior to 2007, the league used the Sony AIBO quadruped robots, Figure B.1(a). Designed to look like a small dog, each robot includes an onboard processor, wireless communication, camera and distance sensors allowing for full autonomy and inter-robot communication with-

---

[1] Known as the Four-Legged League prior to 2007.

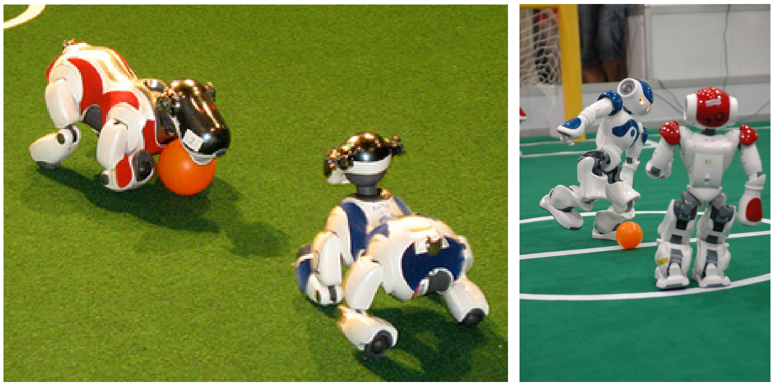

(a) (b)

Figure B.1: Standard Platform League robots: (a) Sony AIBO (b) Aldebaran Nao.

out any input from humans. Beginning in 2007, the league adopted a new robot platform, the Aldebaran Nao humanoid robot, shown in Figure B.1(b). The Nao shares many features with the AIBO, including onboard processing, sensing and communication capabilities, and full autonomy. Its humanoid biped design moves the league closer toward the long-term goal of advanced humanoid robot soccer in 2050.

Soccer in the Standard Platform League is played on a large open field, shown in Figure B.2. All game-related objects in the environment are color coded, including the green field, bi-colored localization beacons, blue and yellow goals, and robot uniform colors. The robots use their onboard camera to identify the color coded objects. Figure B.3(a) shows an example image from an AIBO camera. Before performing object detection, the raw camera image is segmented into color classes, a process during which each image pixel is assigned one of a small set of color class labels. Figure B.3(b) shows the resulting color segmented image, which results in an encoding that requires less memory and processing time to perform further analysis.

The soccer domain presents many real world complexities through a competitive game environment, covering a broad range of AI and robotics research areas, including real time sensor fusion, planning, localization, vision, motion, behavior control and multirobot coordination. During my six years as a developer on the CMU team, including two years as team leader, my work has touched upon all of these areas through a number of projects, including autonomous soccer, autonomous four-legged gait learning [18] and action modeling [19]. Additionally, we have also developed a team of semi-autonomous humanoid game commentators for AIBO soccer games using the QRIO humanoid robots [87]. Figure B.4 presents images of AIBO and Nao soccer games, as well as the humanoid game commentators.

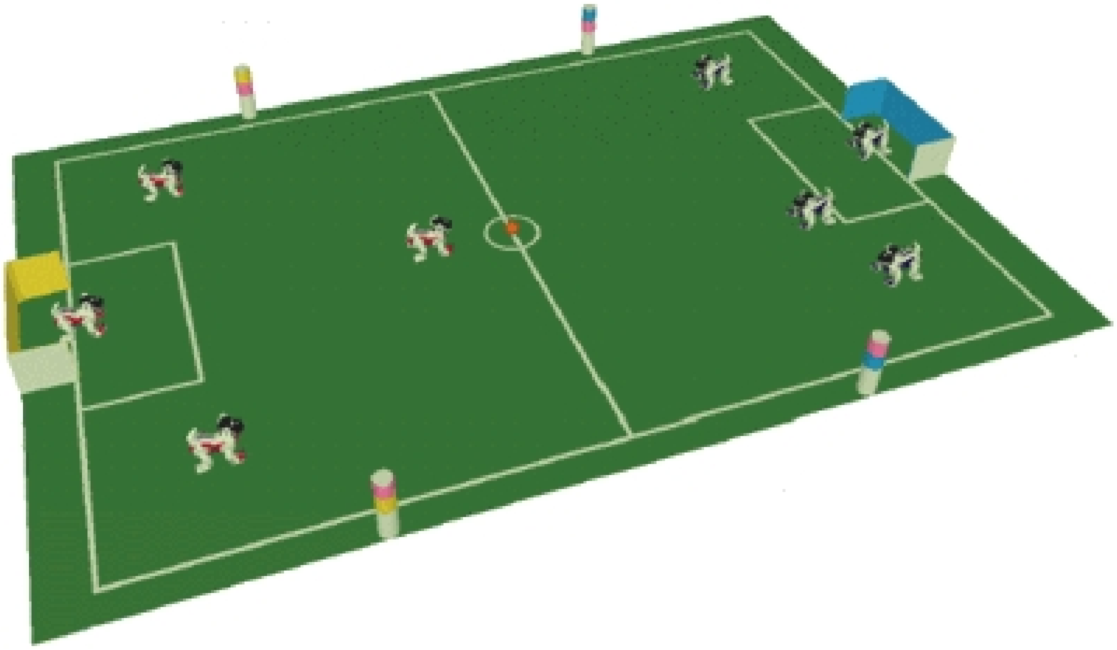
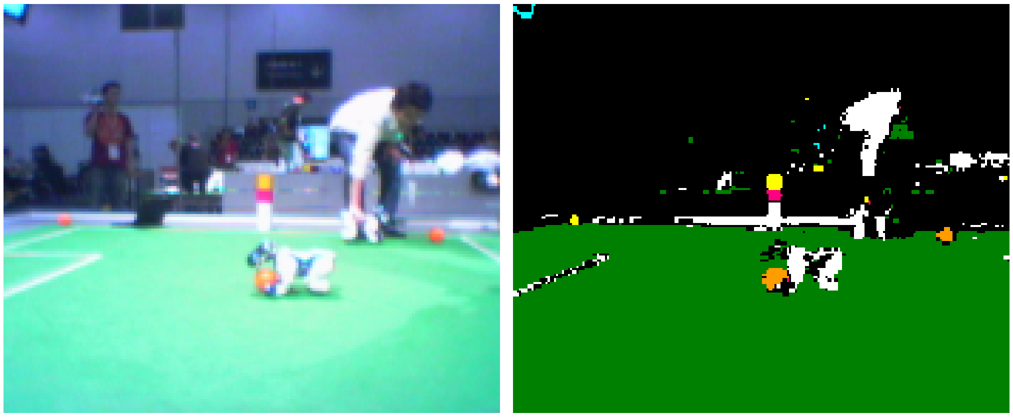
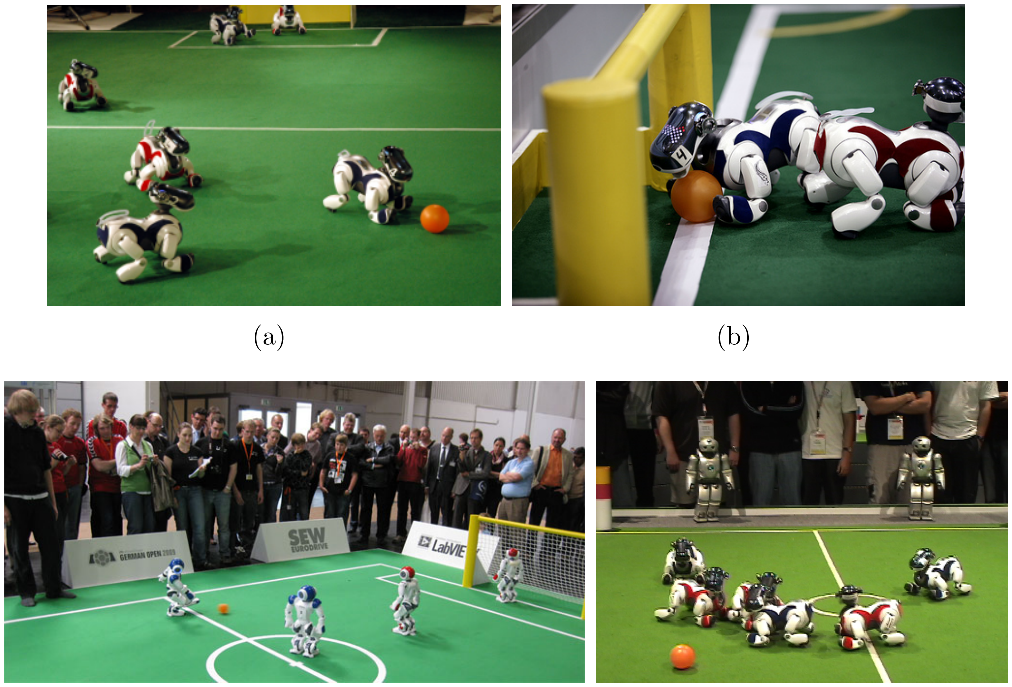


Figure B.2: The field for the 2007 AIBO standard platform league competition.



(a) (b)

Figure B.3: Example vision data from the AIBO camera. (a) The original image. (b) The color segmented image.



(a) (b)

(c) (d)

Figure B.4: Images of robot soccer. (a),(b) AIBO robots playing soccer. (b) Nao robots playing soccer. (c) QRIO robots commentating an AIBO game.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, New York, NY, USA, 2004. ACM Press.

[2] Aris Alissandrakis, Chrystopher L. Nehaniv, and Kerstin Dautenhahn. Synchrony and perception in robotic imitation across embodiments. In *IEEE International Symposium on Computatonal Intelligence in Robotics and Automation*, Kobe, Japan, 2003.

[3] Aris Alissandrakis, Chrystopher L. Nehaniv, and Kerstin Dautenhahn. Towards robot cultures? *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*, 5(1):3–44, 2004.

[4] Brenna Argall, Brett Browning, and Manuela Veloso. Learning from demonstration with the critique of a human teacher. In *Second Annual Conference on Human-Robot Interactions (HRI'07)*, Arlington, Virginia, March 2007.

[5] Brenna Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[6] Minoru Asada, Oliver Obst, Daniel Polani, Brett Browning, Andrea Bonarini, Masahiro Fujit, Thomas Christaller, Tomoichi Takahashi, Satoshi Tadokoro, Elizabeth Sklar, and Gal A. Kaminka. An overview of robocup-2002 fukuoka/busan. *AI Mag.*, 24(2):21–40, 2003.

[7] Christopher G. Atkeson and Stefan Schaal. Robot learning from demonstration. In *International Conference on Machine Learning*, pages 12–20, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[8] Darrin C. Bentivegna, Christopher G. Atkeson, and Gordon Cheng. Learning from observation and practice using primitives. *AAAI Fall Symposium Series, 'Symposium on Real-life Reinforcement Learning'*, 2004.

[9] Aude Billard, Yann Epars, Sylvain Calinon, Gordon Cheng, and Stefan Schaal. Discovering Optimal Imitation Strategies. *Robotics & Autonomous Systems, Special Issue: Robot Learning from Demonstration*, 47(2-3):69–77, 2004.

[10] Avrim Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[11] Cynthia Breazeal, Guy Hoffman, and Andrea Lockerd. Teaching and working with robots as a collaboration. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1030–1037, Washington, DC, USA, 2004. IEEE Computer Society.

[12] Brett Browning, Ling Xu, and Manuela Veloso. Skill acquisition and use for a dynamically-balancing soccer robot. In *Proceedings of Nineteenth National Conference on Artificial Intelligence (AAAI'04)*, 2004.

[13] Sylvain Calinon and Aude Billard. Incremental learning of gestures by imitation in a humanoid robot. In *Second Annual Conference on Human-Robot Interactions (HRI'07)*, Arlington, Virginia, March 2007.

[14] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2007. Software available at http://www.csie.ntu.edu.tw/∼ cjlin/libsvm.

[15] Jason Chen and Alex Zelinsky. Programing by demonstration: Coping with suboptimal teaching actions. *The International Journal of Robotics Research*, 22(5):299–319, 2003.

[16] Jin Chen, Eric Chung, Ross Edwards, Nathan Wong, Eileen Mak, Raymond Sheh, Min Sub Kim, Alex Tang, Nicodemus Sutanto, Bernhard Hengst, Claude Sammut, and William Uther. A description of the rUNSWift 2003 legged robot soccer team. Technical report, University Of New South Wales, 2003.

[17] Sonia Chernova and Ronald C. Arkin. From deliberative to routine behaviors: a cognitively-inspired action selection mechanism for routine behavior capture. *Adaptive Behavior Journal*, 15(2):199–216, 2007.

[18] Sonia Chernova and Manuela Veloso. An evolutionary approach to gait learning for four-legged robots. In *In Proceedings of IROS'04*, September 2004.

[19] Sonia Chernova and Manuela Veloso. Learning and using models of kicking motions for legged robots. In *Proceedings of International Conference on Robotics and Automation (ICRA'04)*, May 2004.

[20] Sonia Chernova and Manuela Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS'07)*, May 2007.

[21] Sonia Chernova and Manuela Veloso. Multi-thresholded approach to demonstration selection for interactive robot learning. In *Proceedings of 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI'08)*, March 2008.

[22] Sonia Chernova and Manuela Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34:1–25, 2009.

[23] John Chilton and Maria Gini. Using the aibos in a cs1 course. In *AAAI Spring Symposium –Robots and Robot Venues: resources for AI education*, Palo Alto, CA, 2007.

[24] Kerstin Dautenhahn Chrystopher Nehaniv. Mapping between dissimilar bodies: Affordances and the algebraic foundations of imitation. In *Second Conference on Autonomous Agents: Workshop on Agents in Interaction - Acquiring Competence through Imitation.*, 1998.

[25] Jeffery Allen Clouse. *On integrating apprentice learning and reinforcement learning*. PhD thesis, University of Massachisetts, Department of Computer Science, 1996. Director-Paul E. Utgoff.

[26] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Mach. Learn.*, 15(2):201–221, 1994.

[27] Rüdiger Dillmann. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems*, 47(2-3):109–116, 2004.

[28] Kevin R. Dixon and Pradeep K. Khosla. Learning by observation with mobile robots: A computational approach. In *in Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.

[29] Iulia Dobai, Leon Rothkrantz, and Charles van der Mast. Personality model for a companion aibo. In *ACE '05: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 438–441, New York, NY, USA, 2005. ACM.

[30] Terrence Fong, Clayton Kunz, Laura M. Hiatt, and Magda Bugajska. The human-robot interaction operating system. In *HRI '06: Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 41–48, New York, NY, USA, 2006.

[31] Holger Friedrich and Rudiger Dillmann. Robot programming based on a single demonstartion and user intensions. In *3rd European Workshop on Learning Robots at ECML'95*, 1995.

[32] Andrew Garland and Neal Lesh. Learning Hierarchical Task Models by Demonstration. Technical Report TR2003-01, Mitsubishi Electric Research Laboratories, 2003.

[33] Dylan F. Glas, Takayuki Kanda, Hiroshi Ishiguro, and Norihiro Hagita. Simultaneous teleoperation of multiple social robots. In *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 311–318, New York, NY, USA, 2008.

[34] Michael A. Goodrich, Timothy W. McLain, Jeffrey D. Anderson, Jisang Sun, and Jacob W. Crandall. Managing autonomy in robot teams: observations from four experiments. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 25–32, New York, NY, USA, 2007.

[35] Daniel H Grollman and Odest Chadwicke Jenkins. Dogged learning for robots. In *IEEE International Conference on Robotics and Automation*, pages 2483–2488, 2007.

[36] Daniel H Grollman and Odest Chadwicke Jenkins. Sparse incremental learning for interactive robot control policy estimation. In *IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA, May 2008.

[37] Gillian Hayes and John Demiris. A robot controller using learning by imitation. In *2nd International Symposium on Intelligent Robotic Systems*, 1994.

[38] Susan G. Hill and Barry Bodt. A field experiment of autonomous mobility: operator workload for one and two robots. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 169–176, New York, NY, USA, 2007.

[39] Gregory Hornby, Seiichi Takamura, Jun Yokono, Osamu Hanagata, Takashi Yamamoto, and Masahiro Fujita. Evolving robust gaits with aibo. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pages 3040–3045, 2000.

[40] Geir Hovland, Pavan Sikka, and Brenan J. McCarragher. Skill acquisition from human demonstration using a hidden markov model. In *IEEE International Conference on Robotics and Automation*, pages 2706–2711, 1996.

[41] Curtis M. Humphrey, Christopher Henk, George Sewell, Brian W. Williams, and Julie A. Adams. Assessing the scalability of a multiple robot interface. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 239–246, New York, NY, USA, 2007.

[42] Tetsunari Inamura, Masayuki Inaba, and Hirochika Inoue. Acquisition of probabilistic behavior decision model based on the interactive teaching method. In *Ninth International Conference on Advanced Robotics (ICAR)*, pages 523–528, 1999.

[43] Ignazio Infantino, Antonio Chella, Haris Dzindo, and Irene Macaluso. A posture sequence learning system for an anthropomorphic robotic hand. *Robotics and Autonomous Systems*, 42:143–152, 2004.

[44] Toru Ishida. Development of a small biped entertainment robot QRIO. In *Micro-Nanomechatronics and Human Science, 2004*, pages 23–28, 2004.

[45] Odest Chadwicke Jenkins and Maja J. Matarić. Performance-derived behavior vocabularies: Data-driven acqusition of skills from motion. *International Journal of Humanoid Robotics*, 1(2):237–288, Jun 2004.

[46] Matthew Johnson and Yiannis Demiris. Perceptual perspective taking and action recognition. *International Journal of Advanced Robotic Systems*, 2(4):301–308, 2005.

[47] Chris Jones, Dylan Shell, Maja Matarić, and B. Gerkey. Principled approaches to the design of multi-robot systems. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, Workshop on Networked Robotics*, 2004.

[48] Daisuke Katagami and Seiji Yamada. Real robot learning with human teaching. In *The 4-th Japan-Australia Joint Workshop on Intelligent and Evolutionary Systems*, pages 263–270, 2001.

[49] A. Kerepesi, E. Kubinyi, G.K. Jonsson, M.S. Magnusson, and . Miklsi. Behavioural comparison of human-animal (dog) and human-robot (aibo) interactions. *Behavioural Processes*, 73(1):92 – 99, 2006.

[50] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawai, and Hitoshi Matsubara. Robocup: A challenge problem for ai and robotics. pages 1–19. 1998.

[51] Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. In *IEEE Transactions on Robotics and Automation*, volume 10, pages 799–822, 1994.

[52] Andrea Lockerd and Cynthia Breazeal. Tutelage and socially guided robot learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[53] Matthew Loper, Odest Chadwicke Jenkins, Nathan Koenig, Sonia Chernova, and Chris Jones. Mobile human-robot teaming with environmental tolerance. In *4th ACM/IEEE International Conference on Human-Robot Interaction*, 2009.

[54] Richard Maclin and Jude W. Shavlik. Creating advice-taking reinforcement learners. *Mach. Learn.*, 22(1-3):251–281, 1996.

[55] Gail F. Melson, Peter H. Kahn, Jr., Alan M. Beck, Batya Friedman, Trace Roberts, and Erik Garrett. Robots as dogs?: children's interactions with the robotic dog aibo and a live australian shepherd. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1649–1652, New York, NY, USA, 2005. ACM.

[56] Andrew W. Moore. Variable resolution dynamic programming: Efefficientlyearning action maps in multivariate real-valued state-spaces. In *Eighth International Workshop on Machine Learning*, pages 333–337, 1991.

[57] Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13(1):103–130, 1993.

[58] Remi Munos and Andrew W. Moore. Variable resolution discretization in optimal control. *Machine Learning*, 49:291 – 323, 2002.

[59] Jun Nakanishi, Jun Morimoto, Gen Endo, Gordon Cheng, Stefan Schaal, and Mitsuo Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47:79–91, 2004.

[60] Ulrich Nehmzow, Otar Akanyeti, Cristoph Weinrich, Theocharis Kyriacou, and Steve A. Billings. Robot programming by demonstration through system identification. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, 2007.

[61] Andrew Y. Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Inverted autonomous helicopter flight via reinforcement learning. In *International Symposium on Experimental Robotics*, 2004.

[62] Monica Nicolette Nicolescu. *A framework for learning from demonstration, generalization and practice in human-robot domains.* PhD thesis, University of Southern California, 2003. Adviser-Maja J. Mataric.

[63] Eugnio Oliveira and Luis Nunes. *Learning by exchanging Advice.* Springer, 2004.

[64] Sascha Ossowski and Ronaldo Menezes. On coordination and its significance to distributed and multi-agent systems: Research articles. *Concurr. Comput. : Pract. Exper.*, 18(4):359–370, 2006.

[65] Vinay Papudesi. Integrating advice with reinforcement learning. Master's thesis, University of Texas at Arlington, 2002.

[66] John Platt. *Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods.* MIT Press, 1999.

[67] Nancy Pollard and Jessica K Hodgins. Generalizing demonstrated manipulation tasks. In *Workshop on the Algorithmic Foundations of Robotics*, December 2002.

[68] Bob Price and Craig Boutilier. Accelerating reinforcement learning through implicit imitation. *J. Artif. Intell. Res. (JAIR)*, 19:569–629, 2003.

[69] Michael J. Quinlan, Stephan K. Chalup, and Richard H. Middleton. Techniques for improving vision and locomotion on the sony aibo robot. In *In Proceedings of the 2003 Australasian Conference on Robotics and Automation*, 2003.

[70] Patrick Riley. *Coaching: Learning and Using Environment and Agent Models for Advice.* PhD thesis, Computer Science Dept., Carnegie Mellon University, 2005. CMU-CS-05-100.

[71] Michael T. Rosenstein and Andrew G. Barto. *Supervised actor-critic reinforcement learning.* John Wiley & Sons, Inc., New York, NY, USA, 2004.

[72] Paul E. Rybski and Manuela M. Veloso. From cmdash'05 to cmrobobits : Transitioning multi-agent research with aibos to the classroom. In *Procedings of the AAAI'05 Mobile Robot Competition and Exhibition Workshop, National Conference on Artificial Intelligence*, Pittsburgh, PA, July 2005.

[73] Paul E. Rybski, Kevin Yoon, Jeremy Stolarz, and Manuela M Veloso. Interactive robot task training through dialog and demonstration. In *HRI'07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 49–56, New York, NY, USA, 2007. ACM.

[74] Joe Saunders, Chrystopher L. Nehaniv, and Kerstin Dautenhahn. Teaching robots by moulding behavior and scaffolding the environment. In *HRI '06: Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 118–125, New York, NY, USA, 2006. ACM Press.

[75] Stefan Schaal. learning from demonstration. In *Advances in neural information processing systems*, pages 1040–1046. MIT press, 1997.

[76] William D. Smart. *Making Reinforcement Learning Work on Real Robots.* PhD thesis, Department of Computer Science, Brown University, Providence, RI, 2002.

[77] William D. Smart and Leslie P. Kaelbling. Effective reinforcement learning for mobile robots. In *IEEE International Conference on Robotics and Automation*, 2002.

[78] William D. Smart and Leslie Pack Kaelbling. Practical reinforcement learning in continuous spaces. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 903–910, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[79] Elisabeth Crawford Sonia Chernova and Manuela Veloso. Acquiring observation models through reverse plan monitoring. In *12th Portuguese Conference on Artificial Intelligence*, December 2005.

[80] Richard S. Sutton and Andew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 1998.

[81] Yasutake Takahashi, Koichi Hikita, and Minoru Asada. A hierarchical multi-module learning system based on self-interpretation of instructions by coach. In *RoboCup 2003: Robot Soccer World Cup VII*, pages 576– 583, 2004.

[82] Andrea L. Thomaz and Cynthia Breazeal. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Twenty-First Conference on Artificial Intelligence (AAAI'06*, 2006.

[83] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

[84] William T. B. Uther and Manuela M. Veloso. Tree based discretization for continuous state space reinforcement learning. In *Artificial Intelligence/Innovative Applications of Artificial Intelligence*, pages 769–774, 1998.

[85] Michael van Lent and John E. Laird. Learning procedural knowledge through observation. In *K-CAP '01: Proceedings of the 1st international conference on Knowledge capture*, pages 179–186, New York, NY, USA, 2001. ACM Press.

[86] Harini Veeraraghavan and Manuela Veloso. Teaching sequential tasks with repetition through demonstration (short paper). In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AMMAS'08)*, May 2008.

[87] Manuela Veloso, Nicholas Armstrong-Crews, Sonia Chernova, Elisabeth Crawford, Colin McMillen, Maayan Roth, Douglas Vail, and Stefan Zickler. A team of humanoid game commentators. *International Journal of Humanoid Robotics*, 2008.

[88] Manuela Veloso, Scott Lenser, Doug Vail, Paul Rybski, Nick Aiwazian, and Sonia Chernova. Cmrobobits: Creating an intelligent aibo robot. In *AAAI Spring Symposium on Accessible, Hands-on AI and Robotics Education*, Palo Alto, CA, 2004.

[89] Richard Voyles and Pradeep Khosla. A multi-agent system for programming robotic agents by human demonstration. In *Proceedings of AI and Manufacturing Research Planning Workshop*, August 1998.

[90] Jijun Wang and Michael Lewis. Human control for cooperating robot teams. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 9–16, New York, NY, USA, 2007.